(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

COMPUTER
BASED
SATELLITE
DESIGN

by

David L. Lashbrook

June, 1992

Thesis Advisor: Brij N. Agrawal

Approved for public release; distribution is unlimited.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE		Form Approved OMB No 0704 0188

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS NONE
2a SECURITY CLASSIFICATION AUTHORITY N/A	3 DISTRIBUTION AVAILABILITY OF REPORT UNLIMITED DISTRIBUATION
2b DECLASSIFICATION DOWNGRADING SCHEDULE N/A	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) Naval Postgraduate School	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, CA. 93943-5000
8a NAME OF FUNDING / SPONSORING ORGANIZATION N/A; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code) N/A	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
COMPUTER BASED SATELLITE DESIGN, UNCLASSIFIED

12 PERSONAL AUTHOR(S)
Lashbrook, David, L.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year Month Day)	15 PAGE COUNT
Master's Thesis	FROM ________ TO ______	1992, MAY 26	235

16 SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB GROUP	

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

A computer program to design geosynchronous spacecraft has been developed. The program consists of four separate but interrelated executable computer programs. The programs are compiled to run on a dos based personnel computer. The source code is written in DoD mandated Ada programming language.
The thesis presents the design technique and design equations used in the program. Detailed analysis is performed in the following areas for both dual spin and three axis stabilized spacecraft configurations: 1) Mass Propellent Budget and Mass Summary 2) Battery Cell and Solar Cell Requirements for a Payload Power Requirement 3) Passive Thermal Control Requirements
Thesis includes a user's manual Appendix A, and the source code for the computer programs as Appendix B.

20 DISTRIBUTION AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a NAME OF RESPONSIBLE INDIVIDUAL Brij N. Agrawal	22b TELEPHONE (Include Area Code) (408) 646-3338	22c OFFICE SYMBOL AA/Ag

DD Form 1473, JUN 86 Previous editions are obsolete SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603

Computer Based Satellite Design

by

David L. Lashbrook
Lieutenant Commander, United States Navy
B.S., University of Missouri

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ASTRONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1992

Author: [illegible]
David L. Lashbrook

Approved by: Brij N. Agrawal
Brij N. Agrawal, Thesis Advisor

[illegible]
Randy L. Wight, Second Reader

[illegible]
Daniel J. Collins, Chairman
Department of Aeronautics and Astronautics

ABSTRACT

A computer program to design geosynchronous spacecraft has been developed. The program consists of four separate but interrelated executable computer programs. The programs are compiled to run on an dos based personnel computer. The source computer code is written in DoD mandated Ada programming language.

The thesis presents the design technique and design equations used in the program. Detailed analysis is performed in the following areas for both dual-spin and three axis stabilized spacecraft configurations:

- Mass Propellent Budget and Mass Summary
- Battery Cell and Solar Cell Requirements for a Payload Power Requirement
- Passive Thermal Control Requirements

Thesis includes a users manual Appendix A, and the source code for the computer programs as Appendix B.

TABLE OF CONTENTS

I. INTRODUCTION

A. PURPOSE

The purpose of this thesis was to design an executahle computer program capable of determining the numbers required to design specified aspects of a geostationary communications satellite, that could be run on a standard home personal computer (PC) The program is based on Professor Brij N. Agrawal's book "Design of Geosynchronous Spacecraft". The book describes all the steps necessary to design a geosynchronous communications satellite. Because of time limitations, it was decided that the following areas of a Geostationary Communications Satellite would be completed on this portion of the project:

- Mass and Propellent Budget
- Electric Power
- Thermal Control

Both Three-Axis and Dual Spin Stabilized satellite configurations are included in the program and provide design reports that can be printed at the user's discretion. Appendix A is a short user's manual of the computer program and Appendix B contains the different executable computer programs' source code.

B. BACKGROUND / CONCEPT

The program came about through discussions with Professor Brij N. Agrawal and space system students enrolled in his satellite design classes. The basis for the computer program is to aid the user in **quickly** determining useful and accurate numbers for a geostationary spacecraft design. Although not a tutorial, the program sequentially walks the user through the necessary steps to develop a Mass Propellent Budget, Photovoltaic Electric Power System, and Thermal Control System for either a three-axis or dual-spin stabilized geostationary communications satellite. The four executable programs will, with a small amount of preparation, allow the average user to develop the described areas of a geosynchronous satellite design in **minutes**. Where as by using a hand calculator, the same variable numbers and final results would take days or weeks to arrive at the same results.

C. OPERATIONAL DESIGN PARAMETERS

All parameters for the executable program used to design a spacecraft in this thesis are based on an orbiting geostationary communications satellite. Some velocity components for propellent usage and therefore mass budget are related to the transfer and parking orbits as well as orbit injection angles and delta velocities necessary to achieve and maintain geostationary orbits. The equations or components used are specifically applicable to a geostationary satellite and might be useful for others as well. The two most common satellite design types, three axis stabilized and dual-spin stabilized, are used as a basis for program engineering and concept.

1. Dual Spin Stabilization

The dual spin configuration used for executable program design is a cylindrical shell with spacecraft power provided by solar cells surface mounted around the outer cylindrical area. The interior of the shell of the cylinder houses spacecraft electronics and propulsion devices.

2. Three Axis Stabilization

Three axis stabilized spacecraft will be based on a rectangular box shape. Power will be provided by sun tracking flat panel array(s). During parking orbit and operational orbit insertion, spin stabilization will be used.

II. MASS SUMMARY AND PROPELLENT BUDGET

This chapter will step through the process of determining a equipment mass summary and a propellent weight budget.

A. TRANSFER ORBIT

For a satellite to be useful it first must achieve its operational orbit. The spacecraft will progress through a series of different orbits before achieving its "final" working orbit. The sequence begins with the launch of the spacecraft from any of a variety of locations depending on payload type, mass, desired orbit inclination, including mission or launch window constraints. The satellite is released from the launch vehicle into a Low Earth Orbit (LEO) parking orbit where system checks are performed. The next phase is to put the satellite into an elliptical transfer orbit in preparation for final insertion into operational orbit.

There are a variety of "transfer" orbits that can be used for final insertion into operational orbit. The most common in-plane and fuel efficient transfer orbit for a geostationary spacecraft is a Hohman Transfer Orbit (HTO). The programs in Appendix B use HTO for velocity change requirements for the transfer orbit. After release from the launch vehicle the spacecraft enters a parking orbit where satellite systems are checked out in preparation for insertion into the final operational orbit.

The first phase of a Hohman transfer is to fire the perigee kick motor (PKM). This places the spacecraft in an elliptical transfer orbit. After four or five transfer orbits the satellite completes system checkout, attitude determinations are finalized, and the apogee kick motor (AKM) fires. Geostationary orbit (GSO) is achieved by inserting the satellite into a circular orbit at a radius of approximately 42,160 kilometers and at a desired inclination (i) of zero degrees (equatorial orbit) for our design purposes. The spacecraft will then go through a series of re-orientations to finalize its operational orbit and mission altitude. (Wertz, 1991, p. 130)

B. VELOCITY

Velocity determination and certain delta velocities required to achieve the operational orbit are vital to building propellent summaries. Other factors taken into consideration are the type of fuel used (relating to I_{sp}) and the efficiencies of the separate orbit insertion and orbit maintenance activities. The efficiencies of the apogee kick motor AKM, PKM, station keeping, and de-orbit functions of the spacecraft must be known so that sufficient propellent, and propellent weight, is incorporated into spacecraft design. Margins for the propellent budget are also assimilated into the propellent budget as a safety factor even though the delta velocity equations provide very accurate figures with little error. All values determined are for a geostationary communication satellite.

1. Geostationary Orbit

In a perfect geostationary orbit the satellite will move around the earth's equator synchronized with the rotation of the earth about the earth's axis. The period (P) is equal to one sidereal day (23 hours, 56 minutes, 4.09 seconds), or from the rising of a star to the rising of the same star vice 24 hours or sunrise to sunrise.

The radius for a geostationary orbit is determined via Keplar's Law using a P of 86,164.09 seconds corresponding to one sidereal day and is given by

$$a=\left(\frac{\mu_e * P^2}{4 * \pi^2}\right)^{1/3} \qquad \text{EQ 2.1}$$

where μ_e= gravitational constant 398,601.2 km^3/(kg/s^2)

a = semi-major axis

P^2= period of the orbit in seconds

The resulting orbital radius is 42,164.2 kilometers and is a pure circular orbit with zero eccentricity.

2. Satellite Drift

Any deviation from the above ideal parameters will cause perturbations in the orbit. The gravitational forces of the sun and moon also act on the satellite to cause drift. Table 2.1 lists the average drift rate imposed on the satellite by the sun and moon.

Table 2.1 SECULAR RATES FROM SUN AND MOON

	Effect of Moon	**Effect of Sun**
Ω	**-0.00076**	**-0.00034**
ω ddot	**$\approx$0.0**	**$\approx$0.0**

a. *East - West Drift*

For a radius greater than (>) 42,164.2 kilometers the spacecraft's orbital period is greater than 24 hours giving it the appearance of drifting westward with respect to the surface of the earth. Conversely, for an orbital radius of less than 42,164.2 kilometers, the spacecraft's orbital period is less than 24 hours, thus giving it the appearance of drifting eastward with respect to the surface of the earth. The incremental drift rate of the longitudinal drift with respect to an ideal geostationary orbit is: (Agrawal, pg 68, 1986)

$$\Delta n = -\frac{3}{2} * \frac{h_s}{a_s} * \Delta a = -\frac{3}{2} * \frac{360^o/day}{42,164.2km} = \frac{-0.013^o/day}{42,164.2km} \qquad \text{EQ 2.2}$$

The east - west motion for an inclined circular orbit is usually very small. This longitudinal drift for a small i and λ is:

$$\lambda = -\frac{i^2}{4} * \sin(2nt) \qquad \text{EQ 2.3}$$

So for small inclinations, latitude oscillation is predominate. This can be easily seen in Figure 2.1. The effects of the sun and moon can be seen in Table 2.1. For an elliptic equatorial orbit with inclination (i=0) the amplitude is

$$\Delta\lambda = 2 * e \qquad \text{EQ 2.4}$$

Therefore, east - west oscillation predominates.

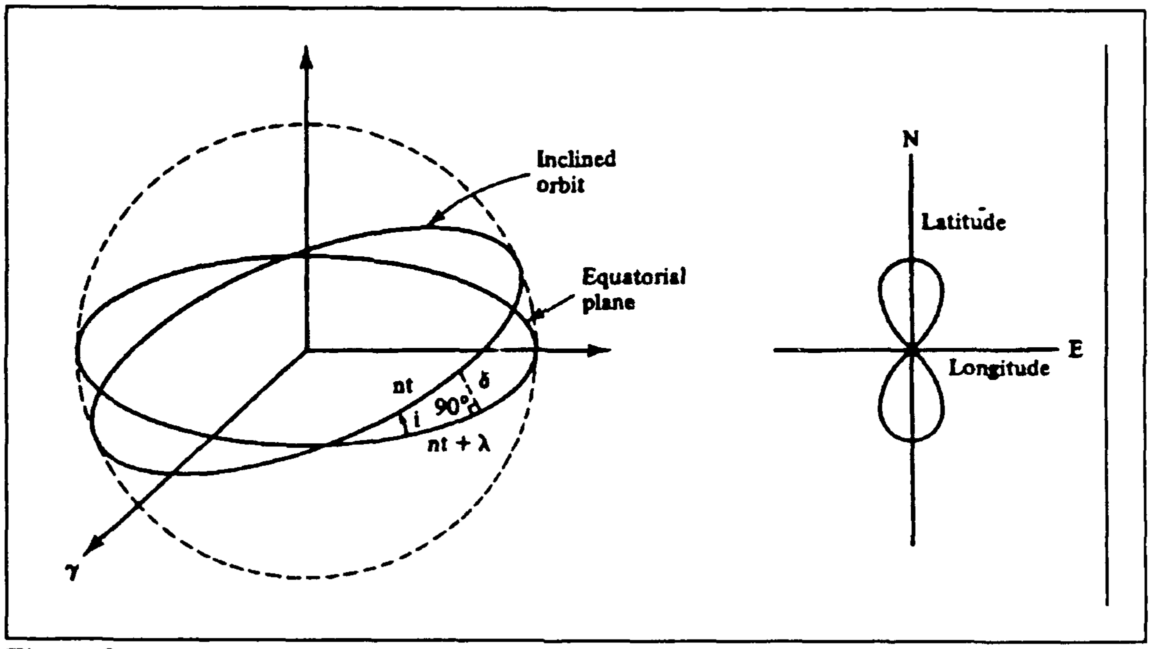

Figure 2.1 Geosynchronous Orbit (i≠0)

b. North South Drift

An inclined geostationary orbit with correct radius will tend to oscillate mainly in latitude with respect to the surface of the earth. The amount of oscillation depends on the inclination and the drift rate. The motion or path of the drift resembles a figure eight (8). See figure 2.1. (Agrawal, p. 88, 1986)

c. Gravitational Effects of Sun and Moon

The gravitational forces of the sun and moon cause secular variations in the orbital elements of a geostationary satellite. These variations have the largest effect on right ascension of the ascending node and the argument of perigee. For nearly circular orbits (one aspect of an ideal geostationary orbit) $e \approx 0$ and the resulting error is on the order of e^2. So the equations for the rates of change from the sun and moon are (Wertz, p. 125, 1991):

$$\dot{\Omega}_{moon} = -0.00138 * \frac{\cos(i)}{n} \qquad \text{EQ 2.5}$$

$$\dot{\Omega}_{sun} = -0.00154 * \frac{\cos(i)}{n} \qquad \text{EQ 2.6}$$

$$\dot{\omega}_{moon} = -0.00169 * (4 - 5 * \sin^2(i)) \qquad \text{EQ 2.7}$$

$$\dot{\omega}_{sun} = 0.00077 * \frac{(4 - 5 * \sin^2(i))}{n} \qquad \text{EQ 2.8}$$

where, i = inclination

n = orbit revolutions per day

For a geostationary orbit, the total longitudinal drift acceleration $\ddot{\lambda}$ is obtained by averaging the drift acceleration over the orbital period. Since the sun for our purposes is an inertially fixed reference point it contributes less than one degree of drift per day. So for a circular orbit the velocity remains constant. Therefore the main contributor to longitudinal perturbations is not the sun or moon but the earth's slightly elliptical shape. This ellipticity caused by what is known as the equatorial bulge changes the earths gravitational force preventing it from acting purely radial. This phenomenon creates two stable longitudes called S1 and S2 (75°E and 255°E) and two unstable longitudes US1 and US2 where the gravitational forces are radial and the longitudinal drift acceleration is zero. Drift will occur whenever the satellite is not at one of these points. For example, if a satellite longitude is between S1 and US1, the lateral force component is along the velocity and this results in a negative longitude acceleration as shown in EQ 2.9. So whenever is near US1 or US2 it will migrate towards the stable longitudes S1 or S2 whichever is closest. (Agrawal, p. 83, 1986)

$$\ddot{\lambda}=-0.0168*\sin^2(\lambda-\lambda_{stable})\,\frac{degrees}{day^2} \qquad \text{EQ 2.9}$$

Taking into account only second order gravity effects the longitudinal drift acceleration is:

where

$\ddot{\lambda}$ = longitudinal drift acceleration in degrees \ day^2

λ = longitude of the satellite in degrees

λ_{stable}= stable longitude of 75°E or 255°E

C. VELOCITY DETERMINATION

Variables from the program strategy that will begin our design are based on a geostationary orbit. First we will assume that some launch vehicle has inserted our spacecraft into a desired parking orbit . For a final GSO, a good parking orbit might be around 6600 kilometers. A parking orbit is a LEO orbit where satellite system checks are performed prior to the other phases of insertion into the final operational orbit (GSO operational orbit $\approx$ 42,160 kilometers).

1. Orbit Insertion Velocities

For the design calculations and for the executable program in appendix B the following equations are used for velocity determination. First, to find the orbit radius the program requests the values for radius at apogee (r_A) and then the value for radius at perigee (r_P) to find the semi-major axis (a).

$$a=\frac{(r_A+r_P)}{2} \qquad \text{EQ 2.10}$$

The equations for transfer orbit velocity at apogee (V_{ta}) and velocity transfer orbit velocity at perigee (V_{tp}) are:

$$V_{parking}=\sqrt{\frac{\mu_e}{a}} \qquad \text{EQ 2.11}$$

$$V_{tp}=\sqrt{\frac{2*\mu_e*r_A}{r_A+r_P}} \qquad \text{EQ 2.12}$$

$$V_{ta}=V_{tp}*\left(\frac{r_P}{r_A}\right) \qquad \text{EQ 2.13}$$

The in-plane velocity change required to transition from parking orbit to transfer orbit is:

$$\Delta V_{transfer}=V_{tp}-V_{Parking} \qquad \text{EQ 2.14}$$

and the mean velocity is given by:

$$V_m=\sqrt{\frac{\mu}{a}} \qquad \text{EQ 2.15}$$

The transfer orbit period is:

$$T_t = 2*\pi*\frac{a^{\frac{3}{2}}}{\mu^{\frac{1}{2}}} = 2*\pi*\sqrt{\frac{\left(\frac{r_A+r_P}{2}\right)^3}{\mu_e}} \qquad \text{EQ 2.16}$$

And ultimately the desired synchronous orbit velocity is:

$$V_s = \sqrt{\frac{\mu_e}{a}} \qquad \text{EQ 2.17}$$

a. Delta Velocity for Apogee Motor Firing

To find the velocity change and insertion angle required to transition from a parking orbit to the final geostationary equatorial orbit used in the program, the orientation of the plane of orbit or inclination must be changed. Figure 2.2 depicts this change. The components of the velocity necessary to achieve equatorial geostationary orbit can be found using the law of cosines. The insertion angle using this law is :

$$\alpha = \arctan\frac{V_{ta}*\sin(i)}{V_s - V_{ta}*\cos(i)} \qquad \text{EQ 2.18}$$

A Hohman Transfer Orbit (HTO) is utilized by the program to determine variable values. The HTO uses a two burn method for insertion into Geostationary orbit. This method is the most fuel efficient and is therefore highly desirable to increase spacecraft life. Using HTO the ΔVelocity for insertion into operational orbit (ΔV_{gt}) is: (Agrawal, p. 95, 1986)

$$\Delta V_{gt}=\sqrt{(V_{ta}*\sin(i))^2+(V_s-V_{ta}*\cos(i))^2} \qquad \text{EQ 2.19}$$

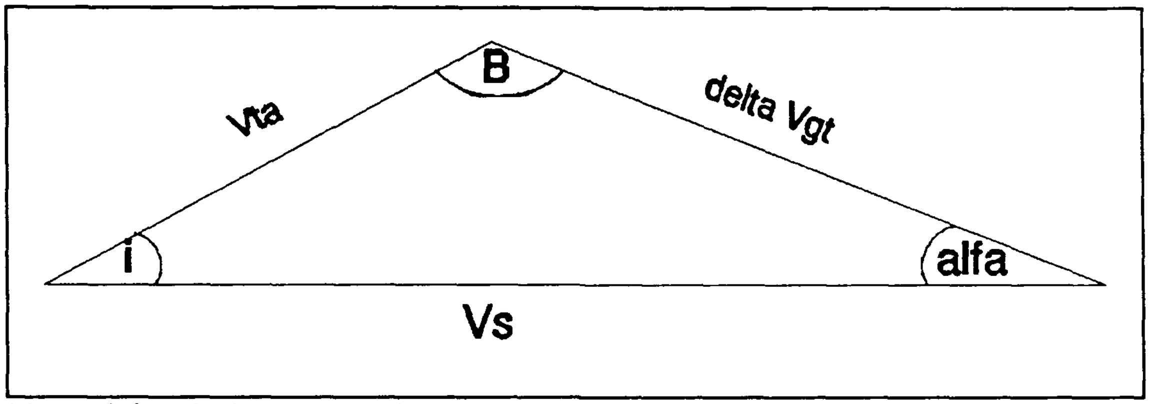

Figure 2.2 Velocity Vector Diagram at Apogee Burn

1. Delta Velocity Station Keeping

The perturbing actions of the sun, moon, and earth's equatorial bulge cause the orbit of a satellite to deviate from the ideal. Table 2.2 lists the different inclination drift rates. Orbital limits are kept within mission parameters by a process called station keeping.

As discussed earlier, for inclination other than zero, drift will be primarily in latitude. Corrections used to keep the spacecraft within the proper operational latitudes is called north-south station keeping. Conversely a change in the ideal geostationary radius will cause a drift in longitude. Station keeping used to maintain the

Table 2.2 INCLINATION DRIFT RATES

Date January 1	Ω_{moon} (Deg)	i_t (Deg)	Ω_{total} (Deg)	i_{dot}moon (Deg)	i_{dot} Total (Deg)
1993	260.526	23.135	-13.023	0.565	0.834
2000	125.177	20.888	11.875	0.523	0.792

longitude within operational parameters is called longitudinal or east-west station keeping.
geostationary radius will cause a drift in longitude. Station keeping used to maintain the longitude within operational parameters is called longitudinal or east-west station keeping.

a. North-South Station Keeping

For a geostationary orbit finding the average inclination drift rate per year (ADPY), including the gravitational effects of the sun, moon, and inclination of the orbit parameters, is found by adding the published yearly drift rates (Table 2.2) during the satellites proposed life and dividing by the spacecraft life. The average time between north-south station keeping is:

$$T_{ns}=\frac{2*i_{tol}}{ADPY}*365.25days \qquad \text{EQ 2.20}$$

where i_{tol} = inclination tolerance

T_{NS} = time spent in north-south station-keeping

The total number of maneuvers ($N_{maneuvers}$) for a given inclination tolerance (i_{TOL})

during the spacecraft life (SL) in years is:

$$N_{maneauvers}=\frac{ADPY*SL}{2*i_{tol}} \qquad \text{EQ 2.21}$$

North-south station keeping is necessary whenever the latitude or inclination drift exceed mission limits. (Wertz, p. 139, 1991)

For north-south drift, the worst case ΔV is

$$\Delta V=6.148*\sin(i_{tol})\left(\frac{kilometers}{second}\right) \qquad \text{EQ 2.22}$$

The contribution of the sun and moon to yearly drift are

$$\Delta V_{moon}=102.67*\cos(\alpha)*\sin(\alpha) \qquad \text{EQ 2.23A}$$

$$\Delta V_{SUN}=40.17*\cos(\gamma)*\sin(\gamma) \qquad \text{EQ 2.23B}$$

where α = angle between the orbital satellite plane and the moons orbital plane

γ = angle between the satellite orbital plane and plane of the eclipticg20

b. North-South Drift

Table 2.3 (Agrawal, 1986, p. 88) gives the average ΔV required for a north south station keeping maneuver, and the time interval between maneuvers for several inclination limits. (Agrawal, 1986, p. 87)

Table 2.3 INCLINATION STATION KEEPING

Inclinations Limit (degrees)	Δ V per Maneuver (m/s)	Average Time Between Maneuvers (days)
0.1	10.7	86.14
0.5	53.65	430.7
1.0	107.30	861.4
2.0	214.56	1722.8
3.0	321.76	2584.2

c. *Longitudinal Station Keeping*

It is realistic to consider only second order effects of the sun and moon on satellite perturbations. In relation to the earth's equatorial bulge, higher order effects are negligible. Table 2.4 lists some ΔV's for different longitudinal tolerances. The longitudinal drift caused by the earth's equatorial bulge is added to these effects giving the longitudinal drift rate. Figure 2.3 illustrates these longitudinal drift oscillations.

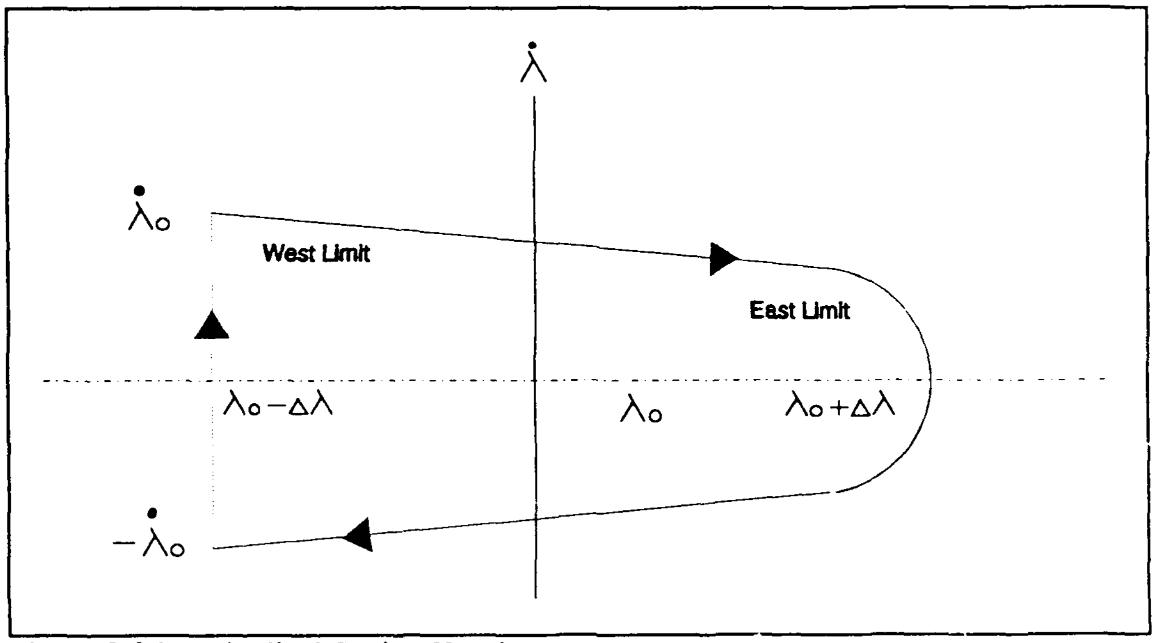

Figure 2.3 Longitudinal Station Keeping

The satellite drift rate is given by

$$\lambda_0=2*(\ddot{\lambda}*\Delta\lambda)^{\frac{1}{2}} \qquad \text{EQ 2.24}$$

Where $\Delta\lambda$ = allowable longitudinal deviation

$\dot{\lambda}_o$ = drift rate when correction is applied (see Figure 2.3)

$\dot{\lambda}$ = longitudinal drift acceleration

The time interval between east-west maneuvers is

$$T_{EW}=4*\left(\frac{\Delta\lambda}{\ddot{\lambda}}\right)^{\frac{1}{2}} \qquad \text{EQ 2.25}$$

where T_{EW} = time spent in east-west station keeping

Table 2.4 LONGITUDE STATION KEEPING

Longitude Tolerance	$\Delta V_{MAXIMUM}$	Minimum Time Interval Between Maneuvers (days)
0.1	0.15	31
0.2	0.21	43
0.5	0.33	69
1.0	0.46	97
2.0	0.66	138
3.0	0.80	169

The velocity change required per year is

$$\Delta V_{year}=5.66*\ddot{\lambda}_o*\frac{365}{T}\frac{ms^{-1}}{year}=1.74*\sin(2*(\lambda-\lambda_{stable})) \quad \text{EQ 2.26}$$

where T = time interval between $\dot{\lambda}_o$ and - $\dot{\lambda}_o$ See Figure 2.3.

Drift acceleration due to the moon is periodic with an approximately 13.6 day cycle. So for all but small longitude tolerances its effects can be neglected. Table 2.4 gives some example ΔV's for different longitude tolerances. (Agrawal, 1986, pp. 89-90)

d. Station Repositioning

At times a satellite may need to move from one station to another this maneuver is called station repositioning. This maneuver is accomplished via two separate velocity changes. The first maneuver uses a velocity change to impart motion in the desired direction, this puts the spacecraft into a slightly elliptic transfer orbit. When the spacecraft nears the desired longitude another velocity change maneuver of equal magnitude but in the opposite direction acts to put the satellite back into a circular synchronous orbit at the new operating longitude. The velocity change (and therefore the fuel required) depends on the amount of time allowed to complete the station change maneuver.

$$\Delta\dot{\lambda}=\frac{\Delta\lambda}{n} \qquad \text{EQ 2.27}$$

where $\Delta\dot{\lambda}$ = required drift rate

$\Delta\lambda$ = longitudinal degrees to be moved

n = numbers of days allowed for movement

The first velocity change required for station repositioning is:

$$\Delta V=2.83*\Delta\dot{\lambda}=2.83*\frac{\Delta\lambda}{n} \qquad \text{EQ 2.28}$$

So the total velocity change (ΔV_{TOTAL}) for starting and stopping the station change is:

$$\Delta V_{TOTAL}=2*\Delta V=5.66*\frac{\Delta\lambda}{n} \qquad \text{EQ 2.29}$$

e. De-Orbit

After a geostationary satellite has finished its useful life it should be "de-orbited" or boosted into a benign orbit to prevent collisions with future active payloads.

Boosting a geostationary satellite into a benign orbit is the most economically feasible method of disposal. The program in appendix B uses a mass ratio based on a reference satellite to determine the approximate fuel required to de-orbit the satellite.

D. MASS PROPELLENT BUDGET

Now that all velocities have been determined and knowing the spacecraft weight and power requirements the weight of the satellite propellent, structural, operational and power systems can be estimated. The propellent budget is made up of four categories:

1) Velocity Control Propellent

2) Attitude Control Propellent

3) Propellent Margin

4) Residual

Attitude control propellent is used for attitude control during ΔV thrusting, spin stabilization, maneuvering while spinning, counter disturbance torques, attitude maneuvering and limit cycling (or oscillation). Propellent margin is a percentage of the propellent requirement and residual is what fuel is left that cannot be used. (Wertz, 1991, p. 262)

1. Propellent Mass

Assuming the mass of the spacecraft is known the satellite propellent mass is: (Agrawal, 1986, p. 45)

$$M_P = M_i * (1 - \exp^{\frac{-\Delta V}{I \cdot g}}) \qquad \text{EQ 2.30}$$

where	M_P	= Mass of the Propellent in Kilograms
	M_i	= Initial Mass of the Spacecraft
	ΔV	= Required Velocity Change to get in operational orbit - m/s
	I	= Specific Impulse of the Fuel
	g	= gravity 9.81 m/s^2

NOTE: The higher the specific impulse (I_{sp}) the lower the amount of fuel necessary to complete satellite mission requirements.

2. Design References

For initial determination of the total fuel requirements, this program develops a mass propellent budget by referencing a known communications satellite with a similar design. The design satellite mass is divided by the reference satellite mass to give a mass ratio (MR), which is used for subsequent calculations.

$$MR = \frac{Design_{SpacecraftMass}}{Reference_{SpacecraftMass}} \qquad \text{EQ 2.31}$$

The next step is to look up the adaptor mass for the launch vehicle our design satellite is using in the applicable launch vehicle users manual. The adaptor mass is then subtracted from the initial spacecraft mass ($M_{i+adaptor}$) to find mass of the spacecraft before apogee burn (M_i), Equation 2.34, this is used to determine the propellent mass for the different mission control events.

$$M_i = M_{i+adaptor} - Mass_{Adaptor} \qquad \text{EQ 2.32}$$

The final variables required by the program are the efficiencies for north-south and east-west station keeping, station repositioning, de-orbit, and apogee injection plus orbit insertion. It is assumed in the executable program that apogee injection and orbit injection efficiencies are the same.

E. MASS DETERMINATION

The pre-amf mass (mass before apogee motor firing) programmed mass is initialized at seven kilograms and is multiplied by the MR (mass ratio) to find the pre-amf MASS for the design.

$$M_{Pre\text{-}Amf} = MR * Pre\text{-}AMF_{Reference} \qquad \text{EQ 2.33}$$

where $M_{Pre\text{-}AMF}$ = Mass of spacecraft before apogee motor firing

$Pre\text{-}AMF_{Reference}$ = Pre - AMF reference Mass

The apogee motor firing mass (M_{AMF}) required to insert the spacecraft into its operational orbit is:

$$M_{AMF} = M_i * (1 - \exp^{\frac{\Delta V_{GT}}{I_{AI} * g}}) \qquad \text{EQ 2.34}$$

where I_{AI} = fuel impulse for the apogee injection motor

ΔV_{GT} = Delta velocity required for geostationary orbit transfer

The Post-amf mass is the mass of fuel required for final despin and stabilization into an operational orbit. For this design, the program initializes the Post-AMF mass at 29.9 kilograms.

$$M_{Post\text{-}Amf} = MR * Post\text{-}AMF_{Reference} \qquad \text{EQ 2.35}$$

where $M_{Post\text{-}AMF}$ = Spacecraft mass after apogee motor firing

$AMF_{Reference}$ = Reference spacecraft post apogee firing mass

Final propellent mass requirements for the proposed design are determined by the level of perceived importance of the individual maneuvers during the spacecraft design life. For this design, the hierarchy is north-south then east-west station keeping, station repositioning, and finally de-orbit.

a. Propellent Mass Station Keeping

The propellant mass required for north-south station keeping during the spacecraft life is given by:

$$M_{NS} = (M_i - M_{AMF} - M_{Pre\text{-}AMF}) * (1 - \exp^{\frac{\Delta V_{NS}}{I_{SK} * g * EFF_{NS}}}) \qquad \text{EQ 2.36}$$

where	M_{NS}	= north south station keeping mass
	ΔV_{NS}	= ΔV required for NS station keeping
	EFF_{NS}	= efficiency
	I_{SK}	= station keeping specific impulse

The propellant mass required for east-west station keeping during the spacecraft life is given by:

$$M_{EW}=(M_i-M_{AMF}-M_{Pre\text{-}AMF}-M_{NS})*(1-\exp^{\frac{\Delta V_{EW}}{I_{SK}*g*EFF_{EW}}}) \qquad \text{EQ 2.37}$$

where	M_{EW}	= east west station keeping mass
	EFF_{EW}	= efficiency

b. Propellent Mass Station Repositioning (M_{SR})

The propellant mass required for station repositioning during the spacecraft life is given by:

$$M_{SR}=(M_i-M_{AMF}-M_{Pre\text{-}AMF}-M_{NS}-M_{EW})*(1-\exp^{\frac{\Delta V_{SR}}{I_{SR}*g*EFF_{SR}}}) \qquad \text{EQ 2.38}$$

where	ΔV_{SR}	= velocity changed required to reposition the spacecraft
	I_{SR}	= Specific impulse of the fuel used to reposition the satellite
	EFF_{SR}	= efficiency of the repositioning maneuver

c. *Propellent Mass De-orbit Control (M_{DE})*

The propellant mass required for de-orbit for the spacecraft is:

$$M_{DE}=(M_i-M_{AMF}-M_{Pre\text{-}AMF}-M_{NS}-M_{EW}-M_{SR})*(1-\exp^{\frac{\Delta V_{DE}}{I_{DE}*g*EFF_{DE}}}) \qquad \text{EQ 2.39}$$

where ΔV_{DE} = velocity changed required to de-orbit the spacecraft

I_{DE} = Specific impulse of the fuel used to de-orbit the satellite

EFF_{DE} = efficiency of the deorbit maneuver

d. *Pressurant Mass (M_{PR})*

The pressurant mass is calculated as the MR multiplied by the reference satellite pressurant mass.

$$M_{PR}=MR*M_{PR\text{-}REF} \qquad \text{EQ 2.40}$$

where $M_{PR\text{-}REF}$ = mass of the pressurant in the reference satellite

e. *Mass Margin (M_M)*

The propellent mass margin (M_M) is the one to two percent safety margin is 10% of the spacecraft dry mass.

$$M_M=(M_{Pre\text{-}AMF}+M_{Post\text{-}AMF}+M_{NS}+M_{EW}+M_{DE}+M_{PR}+M_{SR})*0.02 \qquad \text{EQ 2.41}$$

f. Total Propellent Expenditure (M_{PT})

The total mass of propellent expenditure (M_{PT}) including apogee injection requirements, with a 2% safety margin is:

$$M_{PT}=M_M*51.0+M_{AMF} \qquad \text{EQ 2.42}$$

The dry mass of the unified bi-propellant propulsion M_{UB} which combines the functions of apogee injection, attitude control, and station keeping is: (Agrawal, 1986, p.46)

$$M_{UB}=C_{UB}*M_{PT} \qquad \text{EQ 2.43}$$

where C_{UB} = 0.084 for three axis stabilized

= 0.054 for dual-spin stabilized

To find the total propulsion system mass add M_{PR} and M_{UB} together

F. MASS REQUIREMENTS FOR SATELLITE COMPONENTS

1. Structural Mass (M_{ST})

The mass of the spacecraft structure can only be determined accurately after the final spacecraft configuration and preliminary structural design are completed. However this is usually late in the design process. For a good estimation based on our reference satellite use: (Agrawal, 1986, p. 48)

$$M_{ST} = C_{ST} * M_i \qquad \text{EQ 2.44}$$

where	M_{ST}	= Structural mass in kilograms
	C_{ST}	= 0.087 for three axis stabilized
		= 0.097 for dual-spin stabilized

2. Thermal Control Equipment Mass (M_{TH})

For the program the initial thermal dissipation requirement is unknown. To estimate thermal control mass first we determine the spacecraft beginning of life mass (M_{SBOL}) which is: (Agrawal, 1986, p. 48)

$$M_{SBOL} = M_i - M_{Pre\text{-}AMF} - M_{AMF} - M_{Post\text{-}AMF} - M_{ADAPTOR} \qquad \text{EQ 2.45}$$

Once M_{SBOL} is found, we can use the following equation to find M_{TH}:

$$M_{TH} = C_{TH} * M_{SBOL} \qquad \text{EQ 2.46}$$

where	C_{TH}	= 0.032 for three axis stabilized
		= 0.027 for dual-spin stabilized (Agrawal, 1986, pp. 48-52)

3. Attitude Control System Mass (M_{AC})

The mass for a spacecraft attitude control system depends on many variables. These variables are attitude control, attitude accuracy, amount of redundant safety features, and the size and mass of the spacecraft. For initial estimation purposes for a three axis stabilized system the program uses: (Agrawal, 1986, p. 49)

$$M_{AC}=65+0.022*(M_{SBCL}-700) \qquad \text{EQ 2.47}$$

and for a dual spin stabilized system initial estimate is:

$$M_{AC}=31+0.027*(M_{SBOL}-700) \qquad \text{EQ 2.48}$$

4. Electrical (M_E) and Mechanical System Mass (M_M)

The program in appendix B uses the following electrical and mechanical extrapolations from Agrawal's book "Design of Geosynchronous Spacecraft". M_E is given by: (Agrawal, 1986, p. 52)

$$M_E=0.039*M_{SBOL} \qquad \text{EQ 2.49}$$

and M_M is given by:

$$M_M=0.014*M_{SBOL} \qquad \text{EQ 2.50}$$

5. Mass Margin (M_{Margin})

The mass margin is 10% of the spacecraft dry mass (M_{UB} for a unified bi -propellent system).

$$M_{DRY}=M_i-M_{adaptor}-M_{PR} \qquad \text{EQ 2.51}$$

$$M_{Margin}=M_{DRY}*0.10 \qquad \text{EQ 2.52}$$

6. Propellent Pressurant Mass (M_{PP})

That portion of the propellent tank that pressurizes the fuel tank to maintain fuel flow on demand is: (Agrawal, 1986, p. 53)

$$M_{PP}=M_{PR}-M_{AMF} \qquad \text{EQ 2.53}$$

7. Electric Power System Mass (M_{EL})

The program in appendix B determines the mass of the electric power system based on a reference satellite and design spacecraft power requirements. For the purposes of this design we will assume:

1) 10% margin for the solar array

2) 5% margin for the equipment

3) NiH_2 batteries with spacecraft fully operational during eclipse.

Also the program and design will use the Mass Ratio and the housekeeping power of the reference satellite P_{HK-REF} to find satellite housekeeping power (P_{hk}).

a. *Housekeeping Power (P_{HK})*

$$P_{HK}=MR*P_{HK-REF} \qquad \text{EQ 2.54}$$

Using P_{HK} and the required payload power the battery power load (P_{BL}) with 5% equipment margin is:

$$P_{BL}=(P_{PAYLOAD}+P_{HK})*1.05 \qquad \text{EQ 2.55}$$

and the solar array load (P_{SL}) with 10% margin is:

$$P_{SL}=(P_{PAYLOAD}+P_{HK})*1.1 \qquad \text{EQ 2.56}$$

where $P_{PAYLOAD}$ = Satellite payload power needed in watts

The electrical power system mass is dependent on the power system design. The weights for solar array, charge array, shunt, charge control, battery, and discharge regulator vary between a partially regulated DC bus and a fully regulated DC bus with the later being slightly larger and therefore heavier overall. The total electrical subsystem mass is also greater for higher power requirements and for dual-spin stabilized spacecraft. (Agrawal, 1986, p. 373)

Therefore depending on the spacecraft power requirements and stabilization methods the total mass of the electrical power subsystem is:

$$M_{EL}=(1.05*(P_{Payload}+P_{HK})$$

$$*(1.1*M_{SA}+1.1*M_{CA}+M_S+M_{CC}-M_{BAT}-M_{DR}))*0.001 \qquad \text{EQ 2.57}$$

where	$P_{Payload}$	= Payload power
	M_{SA}	= Solar Array Mass
	P_{SL}	= Solar Array Power
	M_{CA}	= Charge Array Mass
	M_S	= Shunt Mass
	M_{CC}	= Charge Control Mass
	M_{BAT}	= Battery Mass
	M_{DR}	= Discharge Regulator Mass

8. Communications Package Mass (M_{CP})

The communications package is basically the initial spacecraft mass (M_i) minus all the other initial design masses and is given by:

$$M_{CP}=M_i-M_{adaptor}-M_S-M_{TH}-M_{PR}$$

$$-M_{AC}-M_E-M_M-M_{EL}-M_{Margin}-\Delta M_{Post\text{-}AMF}-M_{PP} \qquad \text{EQ 2.58}$$

9. Tracking Telemetry Reference (M_{TT})

The mass of tracking telemetry reference is based on MR and is given by:

$$M_{TT} = MR * M_{TT-REF} \qquad \text{EQ 2.59}$$

III. ELECTRIC POWER

A. INTRODUCTION

The electric power system provides power to the spacecraft during all phases of its life from liftoff to de-orbit. For this phase of the satellite design use a geostationary communications satellite using traveling wave tubes (TWT's) or radio frequency (RF) amplifiers for power transmission of RF energy. Normally, over 80% of the total satellite system power is used by the communications payload. The power remaining is required for spacecraft housekeeping duties.

B. POWER SYSTEM DESIGN

For this design the primary power for spacecraft systems is derived from solar cells. During non-eclipse periods power is provided by solar arrays which convert solar energy via photovoltaic conversion to electric power. A non-eclipse period is when the suns light is shining on the solar cells, an eclipse period is when the sun's rays are not available, for example when a satellite is behind the earth. For eclipse periods, a maximum of 1.2 hours for GSO, power is provided via batteries. The batteries for a standard GSO communications satellite normally are required to provide only partial power during eclipse periods.

The electric power system consists of three parts, the solar array, batteries, and power controı electronics. This design and the program in Appendix B concentrates on the solar array development and battery requirements. The program in Appendix B will determine the solar cell and battery cell needs for any input load for a geostationary three axis or dual-spin stabilized satellite.

1. Design Life Considerations

Power system design begins with the electrical power loads for mission design at beginning of life (BOL) and end of life (EOL). Sometimes EOL power demands are reduced but still allow reduced operational capabilities. For our design and the executable program in Appendix B, we will assume that the satellite mission requires full operational capability for the design life of the spacecraft, or until satellite EOL. Power control electronics and solar panel design will control the smooth operation of the power supply to the payload throughout the life of the spacecraft.

2. Design Spectrum

The program and this design will support the total communications payload and whatever load reduced or full, required during eclipse. Sometimes, load requirements for a broadcasting communications payload during eclipse are lower than full power; however, for military payloads and commercial satellites near continuous full power operation may be a design element. Our design will account for both spectrums of payload power criteria. The required power during eclipse will be a percentage (percent_partial_power) of full bus power. Power control electronics will regulate the spacecraft bus voltage and charge rates but will not be addressed in any detail in this design.

C. BATTERIES

Battery requirements are based on payload power criteria. The design standards for the solar array will hinge on the battery depth of discharge (DOD) and ampere hour (AH) needs during eclipse loading. Also during the launch cycle batteries provide the power for housekeeping functions until the spacecraft is in an operational configuration. The solar arrays during the transfer orbit and while in the parking orbit will supply only intermittent power to the satellite, hence the batteries must make up any difference in load requirements.

1. Energy Storage

All spacecraft that use solar cells for power need some system for energy storage to use during eclipse and peak power demands. Although there are many designs for energy storage, like fuel cells and flywheels, batteries will be used in this design because they are reliable, economical, and a proven technology. Batteries tend to provide stable power for the different operating conditions of a spacecraft's life.

(Agrawal, 1986, p. 356) Nickel Hydrogen batteries will be used in the design and for the executable program in Appendix B.

2. Battery Charging

After the batteries have been used to supply the payload requirements during the parking orbit, transfer orbit, satellite configuration finalization, or eclipse the batteries must be re-charged. This is accomplished through a charge array located on the spacecraft solar panels. The design of the charge array depends on the needed loading during the satellite eclipse period. The batteries normally have more than enough time to recharge before the next eclipse cycle.

3. Battery Requirements

a. Known Variables

Table 3.1 contains the required variables and initialized values for the following calculations. In determining the satellite payload and housekeeping power requirements from calculations in Chapter II, we have the criteria needed to find the battery cell standards.

b. Number of Buses

Normally spacecraft have two buses, each supplying half the total power to the spacecraft loads. This design provides for additional reliability and prevents a single point fault from shutting down the spacecraft and thus allows for some graceful degradation in case of a fault in one of the buses. The program in Appendix B allows for single to multiple bus designs, but favors dual bus configurations.

Table 3.1 VARIABLE NAMES AND SYMBOLS

	3 Axis	Dual-Spin	Variable Name
Minimum Discharge Bus Voltage	28.0	35.0	V_{DB}
Design Satellite Bus Voltage	42.0	50.0	V_{BUS}
Bypass Diode Voltage Drop	1.1	1.1	V_{DD}
EOL Battery Discharge Voltage			V_{EOLBAT}
Eclipse Time	1.2	1.2	T_{ECL}
Depth of Discharge	0.65	0.65	DOD
Maximum Battery Charge Voltage	1.5	1.5	V_{MBC}
Maximum Charge Voltage			V_{BC}
Number Series Connected Diodes	3	3	N_{SD}
Battery Charger Voltage Drop	1.75	1.75	V_{CD}
Charge Discharge Voltage Drop	1.1	1.1	V_D
Charge Discharge Efficiency Battery	0.9	0.9	n_{CD}

c. ˙ ˙mber of Battery Cells

The number of battery cells (N) needed to maintain a stated minimum discharge bus voltage (V_{DB}) is determined via manipulation of equation 3.1. The power, voltage times current, required by the bus will be all or part of the total power, for example if the total power required is 1000 watts and the spacecraft has two buses then each bus requires 500 watts of power. Therefore batteries for each bus must provide this power for the maximum eclipse time of 1.2 hours for a GSO orbit.

$$V_{DB}=(N-1)*V_D-V_{DD} \qquad \text{EQ 3.1}$$

therefore N is

$$N=\frac{V_{DB}+V_{DD}}{V_D}+1 \qquad \text{EQ 3.2}$$

then round N up to the next higher integer and using this new value find the new minimum discharge bus voltage V_{DB}.

d. Minimum Discharge Bus Voltage (V_{DB})

For electrical equipment to operate correctly a minimum voltage V_{DB} must be available and is given by:

$$V_{DB}=(N-1)*V_D-V_{DD} \qquad \text{EQ 3.3}$$

e. Battery Cell Ampere Hours

The battery cell ampere hour ($CELL_{AH}$) requirement is the number of ampere hours needed to supply the satellite during eclipse and is determined from the following equation:

$$CELL_{AH}=\frac{\frac{P_{BL}}{N_{BUSES}}*T_{ECL}}{V_{DB}*DOD} \qquad \text{EQ 3.4}$$

where	N_{BUSES}	= number of buses
	T_{ECL}	= maximum time in eclipse
	P_{BUS}	= power in watts per bus
	DOD	= depth of discharge

and the bus power (P_{BUS}) requirements are:

$$P_{BUS}=\frac{P_{TOTAL}}{N_{BUSES}} \qquad \text{EQ 3.5}$$

f. Maximum Battery Charge Voltage

The maximum battery charge voltage (V_{BC}) assuming an open circuit failure of one battery cell is:

$$V_{BC}=V_{MBC}*(N-1)+N_{SD}*V_{SD} \qquad \text{EQ 3.6}$$

where V_{MBC} = allowable battery charge voltage

N_{SD} = number of series connected diodes

V_{SD} = voltage drop across each series connected diode

g. Bus Voltage Allowable Deviation

Usually the bus voltage V_{BUS} is allowed to vary within prescribed limits of $\approx \pm 0.5$ volts. This value is called the bus voltage allowable deviation (V_{DEV}) and it allows us to calculate(V_{BUSLL}) the lower limit of bus voltage which is:

$$V_{BUSLL} = V_{BUS} - V_{DEV} \qquad \text{EQ 3.7}$$

h. Voltage Charge Array

The required boost voltage needed by the charge array (V_{CA}) is given by:

$$V_{CA} = V_{BC} - V_{BUS} + V_{CD} \qquad \text{EQ 3.8}$$

i. Seasonal Currents

Design parameters for the program in Appendix B provides for a charge current to be applied to each bus as a percentage of the number of buses. For one bus it would be a 100% duty cycle. For two buses, the charge current would be applied to each bus on a 50% duty cycle.

The charge rate current for autumnal equinox $I_{EQUINOX}$ is given by:

$$I_{EQUINOX}=\frac{CELL_{AH}}{15} \qquad \text{EQ 3.9}$$

and for the summer solstice current $I_{SOLSTICE}$ it is:

$$I_{SOLSTICE}=\frac{CELL_{AH}}{45} \qquad \text{EQ 3.10}$$

Battery cells, in this design, charge at a high rate to return the energy depleted during eclipse to each battery.

j. Battery Recharge Power

The power needed to recharge the batteries at equinox (P_{EC}) is:

$$P_{EC}=V_{BC}*I_{EQUINOX} \qquad \text{EQ 3.11}$$

and the power needed to recharge at solstice (P_{SC}) is

$$P_{SC}=V_{BC}*I_{SOLSTICE} \qquad \text{EQ 3.12}$$

k. Longest Battery Recharge Time

The time to fully recharge the batteries ($T_{RECHARGE}$) is:

$$T_{RECHARGE}=\frac{\frac{P_{BL}}{N_{BUSES}}*T_{ECL}}{P_{EC}*n_{CD}} \qquad \text{EQ 3.13}$$

The batteries will remain on trickle charge for the remainder of the non-eclipse period.

D. SOLAR ARRAY DESIGN LOAD

The solar array design load is the summation of the equipment load and the power required for charging the batteries. Taking the 10% design margin into account the solar array design load at equinox (P_{SALEQU}) is:

$$P_{SALEQU} = (P_{BUS} + P_{EC}) * 1.1 \qquad \text{EQ 3.14}$$

and the solar array design load at solstice (P_{SALSOL}) is:

$$P_{SALSOL} = (P_{BUS} + P_{SC}) * 1.1 \qquad \text{EQ 3.15}$$

This design process and the program in Appendix B will split the loads among the buses if there is more than one. The solar arrays are assumed to operate at the maximum power point of the solar cell "IV" curve at the spacecraft EOL.

E. SOLAR CELLS

The total load on the solar array will be the summation of the equipment load and the power required for charging the batteries. Generally, and for this design and the program in Appendix B, a 10% design margin is used to take into account the uncertainty in the degradation due to radiation and other design factors of the solar array. Equations 3.14 and 3.15 account for this design margin. The next step is to determine the total number of solar cells needed to meet all of the mission requirements including all previous assumptions.

1. Cell Variables

The following variables listed in Table 3.2 and Table 3.3 variables for the particular solar cell used need to be known to begin solar array power system design.

Table 3.2 SOLAR CELL VARIABLES FOR I AND CELL DIMENSIONS

VARIABLE	SYMBOL
Cell Width	C_W
Cell Length	C_L
Cell Thickness	C_T
CURRENT	
Solar Intensity	S
Assembly Losses Current	K^I_A
Environmental Degradation in Current	K^I_D
Solar Intensity Factor including Incidence Angle	K_S
Temperature Coefficient for Current	α_I
Solar Maximum Power Point EOL Summer Solstice	I
Solar Cell Current BOL	I_{mp}

Table 3.3 SOLAR CELL VARIABLES VOLTAGE AND ILLUMINATION

Voltage	
VARIABLE	**SYMBOL**
Panel Wiring Loss per Cell	ΔV
Solar Cell Voltage at Maximum Power Point, BOL	α_V
Radiation Degradation Factor for Voltage	K^V_E
Solar Cell Voltage at EOL	V
Solar Cell Voltage at Maximum Power Point, BOL	V_{mp}
Operating Temperature	T
Test Temperature	T_T
EFFECTIVE ILLUMINATION	
Sun Tracking Flat Panel	1
Dual Spin Surface Mounted	$1/\pi$

2. Parallel Solar Cells (N_P)

Using the variable values from Tables 3.2 and 3.3 and taking the design factors into account, the solar cell current (I) at EOL summer Solstice (T ≈ 39°C) is:

$$I=(I_{mp}+\alpha_I*(T-T_T))*K^I{}_A*K^I{}_D*K_S \qquad \text{EQ 3.16}$$

The total current (I_T) per bus or wing is:

$$I_T=\frac{P_{BUS}}{V_{BUS}} \qquad \text{EQ 3.17}$$

This gives us the total number of solar cells needed in parallel (N_P) for each wing which is:

$$N_P=\frac{I_T}{I} \qquad \text{EQ 3.18}$$

The number for N_P is then raised to the next higher integer number value. For example, 51.135 would be raised to 52 for future calculations using N_P, to give the total N_P for that array or bus. Obviously, the more parallel strings of solar cells the more current is supplied to the satellite bus.

3. Series Solar Cells (N_S)

To find the number of solar cells in series necessary to supply the required bus voltage (V_{BUS}) we first determine the solar cell voltage (V) at EOL summer solstice, which is given by:

$$V=((V_{mp}-\Delta V+\alpha_V*(T-T_T))*K^V{}_E \qquad \text{EQ 3.19}$$

The number of solar cells in series (N_S) needed to supply this voltage, taking into account the voltage drop in the bus ($V_{BUS\text{-}DROP}$), and the voltage drops in the spacecraft wiring harness (V_{WH}) and slip rings (V_{SR}), is:

$$N_S=\frac{V_{BUS}+V_{BUS\text{-}DROP}+V_{DROP\text{-}WH}+V_{DROP\text{-}SR}}{V} \qquad \text{EQ 3.20}$$

Like N_P , N_S is raised to the next higher integer value.

4. Solar Cell Current and Voltage EOL Equinox

At EOL autumnal equinox the solar cell current I_{EQ} and voltage V_{EQ} are given by equations 3.16 and 3.19 respectively, except with T = temperature at equinox ($\approx$49°C).

F. CHARGE ARRAY DESIGN

The charge array supplies the boost voltage necessary to recharge the spacecraft batteries after eclipse.

1. Series Solar Cells Charge Array (N_C)

The number of series solar cells (N_C) required to supply the needed boost voltage is

$$N_C = \frac{V_{CA}}{V} \qquad \text{EQ 3.21}$$

where the voltage (V) is the voltage at solstice.

2. Parallel Solar Cells Charge Array Solstice (N_{SSC})

The number of solar cells needed in parallel for the charge array during summer solstice (N_{SSC}) is

$$N_{SSC} = \frac{CELL_{AH}}{I_{SOLSTICE}} \qquad \text{EQ 3.22}$$

and for equinox the solar cells required in parallel (N_{EQC}) is

$$N_{EQC} = \frac{CELL_{AH}}{I_{EQUINOX}} \qquad \text{EQ 3.23}$$

So the charge array would require N_C solar cells in series and N_{SSC} cells in parallel for summer solstice battery charging, and N_C solar cells in series and N_{EQC} cells in parallel for autumnal equinox battery charging. For a spin stabilized spacecraft the solar cell requirements would have to be multiplied by a factor of π to account for the lower illumination factor.

IV. THERMAL CONTROL

A. BACKGROUND

Spacecraft in all orbits must dissipate heat. Heat can be absorbed from sunlight, reflected sunlight, and planet emitted radiation. (Agrawal, 1986, p. 280) Heat is also generated within the satellite by communication transmitters, batteries, control elements in the power system, the payload, and when apogee kick motors or engines are fired, heat is radiated from their components. (Wertz, 1991, p. 370)

A thermal control system must maintain equipment within an allowable temperature range for optimum operational capability while maintaining an economical design. Spacecraft electronics usually have temperature limits between 0°C and 40°C, while batteries will have limits between 0°C and 20°C. Silicon solar cells operate between +100°C and -100°C, but operate most efficiently at the lower end of this range. (Wertz, 1991, p. 370)

The thermal control system acts upon and is acted upon by almost every other spacecraft system while maintaining the spacecraft systems within their operational temperature parameters. This is especially true of the spacecraft power system because the thermal control system must dissipate all its excess energy and radiate this energy to space.

Though there are many types of thermal control systems from passive and semi-passive to active; this design will use passive techniques. Passive systems have no moving parts or heaters and rely on paints, second surface mirrors, multi-layer insulation, phase changing devices, and radiation or conduction to space radiators like optical solar reflectors (OSR). In addition, passive thermal control designs are usually lighter, cost less, and use less power than active control systems.
(Wertz, 1991, p. 371)

B. THERMAL RADIATOR SIZING

1. Variable Definitions

Table 4.1 lists variable definitions and applicable numerical values.

Table 4.1 VARIABLE DEFINITIONS AND VALUES

VARIABLE DESCRIPTION	SYMBOL \ NUMERICAL VALUE
Thermal Dissipation [W]	P
Emittance of Radiator	ϵ
Radiator Temperature	T \ ($\approx$ 310°K)
Equilibrium Temperature	T_{EQUIL}
Solar Array Diameter	D_{SA}
Radiator Height	H_{RAD}
Solar Absorbtance EOL	α_S
Solar Intensity Winter Solstice	S \ 1397 [W/m^2]
Solar Intensity Summer Solstice	S \ 1311 [W/m^2]
Solar Intensity Vernal Equinox	S \ 1362 [W/m^2]
Solar Intensity Autumnal Equinox	S \ 1345 [W/m^2]
Solar Aspect at Winter Solstice	Θ \ 23.5°
Solar Aspect at Autumnal Equinox	Θ \ 0°
Stefan Boltzmann	σ \ 5.67*10^{-8} [W/(m^2*K^4]
Efficiency	n

2. Thermal Dissipation (P)

Thermal radiators are used to dissipate excess heat, the more heat that must be dissipated the greater the thermal radiator area required. For this design we will assume that the radiator is isothermal and will be sized for the greatest thermal dissipation required. The hottest time is at vernal equinox when the solar intensity is 1362 W/m^2 with a solar aspect of 0°. The thermal dissipation (P) required of the thermal radiator(s) is based on the percentage of the payload power (P_{PP}) and the percentage of housekeeping power (P_{PH}) required to be dissipated as heat. The thermal dissipation is given by:

$$P=\frac{1}{N_{RADIATING\text{-}FACES}}*(P_{PAYLOAD}*P_{PP}+P_{HK}*P_{PH}) \qquad \text{EQ 4.1}$$

Where	P_{HK}	= housekeeping power
	$P_{PAYLOAD}$	= payload power
	$N_{RADIATING\text{-}FACES}$	= number of thermal radiating faces

3. Radiator Area

Radiator area (A) for a three axis stabilized spacecraft. is given by

$$A=\frac{P}{\epsilon*\sigma*T^4*n-\alpha_S*S*\sin(\Theta)} \qquad \text{EQ 4.2}$$

For a dual spin stabilized spacecraft we find the radiator height (H_{RAD}) and multiply it by the circumference of its cylindrical shell structure. The height of the radiator is

$$H_{RAD}=\frac{P}{(D_{SA}*(\pi*\sigma*\epsilon*T^4*n)-(\alpha_S*S*\cos(\Theta))} \qquad \text{EQ 4.3}$$

C. EQUINOX TEMPERATURE

1. Non Eclipse

For design purposes and the program in Appendix B the temperature at equinox ($T_{EQUINOX}$) is needed for both full and partial power requirements during equinox. Temperature after equinox in °K for a three axis stabilized spacecraft is:

$$T_{EQUINOX}=\left(\frac{P}{\epsilon*\sigma*n*A}\right)^{\frac{1}{4}} \qquad \text{EQ 4.4}$$

Temperature after equinox in °K for a dual spin stabilized satellite is:

$$T_{EQUINOX}=\left(\frac{P}{D_{SA}*H_{RAD}*\pi*\epsilon*\sigma*n*A}\right)^{\frac{1}{4}} \qquad \text{EQ 4.5}$$

To determine the temperature in °C subtract 273.15° from the values in equations 4.4 and 4.5. As stated earlier, the temperature range of the spacecraft components (other than solar cells) should be kept between 0°-40°C. If a condition arises where this is not possible to keep the temperature within operational ranges passively, then auxiliary heaters will be required to prevent damage to the spacecraft components.

If batteries are providing full power during eclipse, the radiator temperature will remain approximately the same during eclipse as during the non-eclipse period of equinox. (Agrawal, 1986, p. 284)

2. Eclipse

When batteries provide partial power *(a variable is known as percent partial power (PCT_{PP}))*, heat dissipation is a percentage of thermal dissipation (P) and the equilibrium temperature during equinox (T_{EQUIL}) is

$$T_{EQUIL} = \left(\frac{PCT_{PP} * P}{\epsilon * \sigma * A}\right)^{\frac{1}{4}} \qquad \text{EQ 4.6}$$

and the time constant (τ) for a three axis stabilized spacecraft is

$$\tau = \frac{m * C_P}{4 * \epsilon * \sigma * A * T_{EQUIL}^3} \qquad \text{EQ 4.7}$$

where m = mass of the radiator plus mounted equipment.

For a dual spin stabilized spacecraft, the time constant (τ) is:

$$\tau = \frac{m * C_P}{4 * \epsilon * \sigma * \Pi * D_{SA} * H_{RAD} * T_{EQUIL}{}^3} \qquad \text{EQ 4.8}$$

D. RADIATOR TEMPERATURE AT END OF ECLIPSE

To find the radiator temperature at the end of eclipse this design uses an iterative process. Knowing the maximum time for eclipse we can iterate repetitively through equation 4.9, which is known as the radiative cooling equation, guessing different temperatures until we get a time very close to our maximum eclipse time (T_{EC}) of 1.2 hours (4320 seconds). We then bracket the maximum eclipse time of 72 minutes, with a value close enough to 72 minutes to make the results accurate, (1.2 hours or 4320 seconds) and interpolate to find the minimum temperature during eclipse.

$$\frac{T_{EC}}{\tau} + C = 2 * (\coth^{-1}(\frac{T_{EQUINOX}}{T_{EQUIL}}) - \cot^{-1}(\frac{T_{EQUINOX}}{T_{EQUIL}})) \qquad \text{EQ 4.9}$$

First we must find the constant "C" assuming t=0, where "C" is some constant.

$$C=2*(\coth^{-1}(\frac{T_{EQUINOX}}{T_{EQUIL}})-\cot^{-1}(\frac{T_{EQUINOX}}{T_{EQUIL}})) \qquad \text{EQ 4.10}$$

E. RADIATOR MINIMUM OPERATING TEMPERATURE

Now it is desirable to determine the time needed for the radiator to reach it's minimum operating temperature. First find we must the constant "C" for equation 4.11 , known as the radiative heating equation,by assuming $T_{BC}=0$, and substituting it into Equation 4.11.

$$\frac{T_{BC}}{\tau}+C=2*(\tanh^{-1}(\frac{T_{EQUINOX}}{T_{EQUIL}})-\tan^{-1}(\frac{T_{EQUINOX}}{T_{EQUIL}})) \qquad \text{EQ 4.11}$$

this gives us:

$$C=2*(\tanh^{-1}(\frac{T_{EQUINOX}}{T_{EQUIL}})-\tan^{-1}(\frac{T_{EQUINOX}}{T_{EQUIL}})) \qquad \text{EQ 4.12}$$

where	$T_{EQUINOX}$	=	lowest temperature during equinox as iterated via eq. 4.10
	T_{EQUIL}	=	Equilibrium for the thermal dissipation (P) desired eclipse

To find the time needed to reach a desired radiator operating temperature (T_{RAD}) we substitute the new after equinox temperature values and the time to reach T_{RAD} is now given by

$$t=(2*(\tanh^{-1}(\frac{T_{EQUINOX}}{T_{EQUIL}})-\tan^{-1}(\frac{T_{EQUINOX}}{T_{EQUIL}})))-C)*\tau \qquad \text{EQ 4.13}$$

where t = time to reach specified radiator operating temperature

F. SOLAR ARRAY TEMPERATURE

1. Variable Names and Definitions

Table 4.2 lists the required variables and their symbolic representations for determining the solar array operating temperature.

2. Solar Array Operating Temperature

In a solar array only a fraction of the solar flux is converted into electric power by the solar cells thus reducing the heat of the array. This reduction in the solar absorbtance is called the effective solar absorbtance and it is a function of the average solar cell array absorptance, the packing factor, and the cell efficiency. The solar array operating temperature (T_{OP}) is calculated by first determining the effective solar absorbtance (α_{SE}) which is: (Agrawal,1986, p. 285)

$$\alpha_{SE}=\alpha_S-F_P*\dot{\eta} \qquad \text{EQ 4.'4}$$

Then the operating temperature for any season can be determined by substituting the appropriate solar intensity (S) and solar flux incidence angle (θ)

Table 4.2 VARIABLE NAMES AND DEFINITIONS

VARIABLE / CONSTANTS	SYMBOL
Effective Solar Absorbtance	α_{SE}
Average Solar Cell Absorbtance	α_S
Solar Cell Packing Factor	F_p
Solar Cell Operating Efficiency	η
Array Front Side Area	A_F
Array Back Side Area	A_B
Emittance of the Array Front Side	ϵ_F
Emittance of the Array Back Side	ϵ_B
Solar Intensity	S
Stefan Boltzmann	σ
Angle of Incidence of Sunlight	θ

V. CONCLUSION

The executable programs and technical documentation devised by this thesis allow for the accurate and timely calculation of spacecraft design requirements. In addition, it enables the average user to have the time to explore the different phases of design in more detail. The programs in Appendix B expeditiously calculate the spacecraft mass propellent budget, spacecraft mass summary, solar power system design, and passive thermal control design for a geostationary communications satellite. The results are compiled in an ascii based text file that can be printed and available for user study.

The Ada programming language was used due to its unique ability to write source code in plain english. The programs are written so that future developers with limited programming experience can understand the progress of the program and use it as a starting point for future research.

Future research and development is needed for computer based design in the following geosynchronous areas:

- Spacecraft Attitude Control Systems
- Spacecraft Structural Design

For low earth orbits (LEO) development is desired in:

- Spacecraft Mass Propellent Budgets
- Spacecraft Electric Power System Design
- Spacecraft Attitude Control
- Spacecraft Thermal Control Systems

LIST OF REFERENCES

1. **Agrawal, Brij N.**, *Design of Geosynchronous Spacecraft*, Prentice Hall, Inc., 1986

2. **Wertz, James R., Larson, Wiley J.**, *Space Mission Analysis and Design* , Kluwer Academic Publishers, 1991

APPENDIX A

COMPUTER BASED SATELLITE DESIGN

USERS MANUAL

A. THERMAL

1) To start the program type **"THERMAL"** at the dos prompt.

2) The program will ask "Is your spacecraft spin stabilized". If it is answer with a character 'Y' if not answer with a character 'N'

3) Program will state design chosen

a) "Spacecraft is Spin Stabilized" or

b) "Spacecraft is Three Axis Stabilized"

4) Program asks "What is your Spacecraft Mass"

a) Enter Spacecraft Mass including the Adaptor Mass in kilograms. Example: "3000.0"

5) Program asks "Enter the POWER requirements of the Spacecraft in watts. Example: "2000.0"

6) Next the program asks you to chose a reference satellite from those listed

'1' Intelsat V

'2' Intelsat VI

'3' Intelsat VII

'4' or to insert your own values.

7) The program lists the calculated housekeeping power and asks if you want to change this value for future calculations. If you want to change it enter a character 'Y' if not, depress a character 'N'.

Program then lists the values for Payload Power and Housekeeping Power

8) Program lists some default values for:

$\grave{\eta}$ - efficiency

α - solar aspect coefficient

S - solar intensity solstice

S - solar intensity equinox

$T_{RADIATOR}$ - radiator temperature

$\epsilon_{RADIATOR}$ - radiator emissivity

$T_{ABSOLUTE}$ - absolute zero

$T_{ECLIPSE}$ - eclipse time

$N_{THERMAL\ EMITTING\ FACES}$ - number of thermal emitting faces

$M_{RADIATOR\ PLUS\ EQUIPMENT}$ - mass of radiator plus equipment

P_{PP} - Percent partial power (percent of payload dissipated as heat)

C_P - specific heat

To change any of the listed values enter a character 'Y' and then enter the number of the value you wish to change.

9) Next an informational screen is displayed that lists the different passive thermal control materials and there typical application.

10) Program displays a screen that lets you pick the type of material to use for thermal control. Simply enter the integer of the material desired or enter "10" to enter your own values. Chosen values are then displayed.

11) Program asks user to enter "Percent Payload Power that must be dissipated as heat"

and to enter "Percent Housekeeping Power that must be dissipated as heat"

Example: 33.4% would be entered as "0.334"

Total required power dissipation is then displayed.

12) Program asks user to enter "Solar Array Diameter in meters" if satellite is Dual Spin Stabilized Example "3.456"

13) Program lists "Radiator Height" if satellite is Dual Spin Stabilized.

14) Program asks user to enter a value for "Radiator Efficiency".

The only reason this is in the program is that some users expressed a desire to change the efficiency on the run.

15) Program lists values for:

Temperature at Equinox
Equilibrium Temperature (based on partial power needed during eclipse)
Time Constant in seconds and minutes

16) Program then determines the constant for radiative cooling based on the temperature at equinox and equilibrium temperature. The next step is finding the "After Equinox Temperature" via the radiative cooling equation. The program goes through a series of questions to aid the user in finding the "After Equinox Temperature".

The program outputs your input temperature in °K and the "Eclipse Time" in minutes. Since we know for a geosynchronous orbit the longest eclipse time is 72 minutes we will bracket this time (one value above and one value below 72 minutes). When a temperature yields a value sufficiently close to 72 minutes (around 10 minutes either side) we accept that value by answering with a character 'Y' and that value is saved to bracket one side of 72. Next the program lists the saved "After Equinox Temperature" and its associated time. Then the program then goes the same process again to bracket the other side of 72 minutes. The program will not allow the user to accept a second on the same side of 72 minutes as the first. Once 72 minutes has been successfully bracketed the program interpolates to determine the "Temperature After Equinox" for a time of 72 minutes.

17) The program then moves on to radiative heating portion, the listed default desired operating temperature is listed and the user is asked if he wishes to change the value for either the "Radiator Heat Dissipation" or the "Specified Operating Temperature".

The following values are listed based on user input:

Equilibrium Temperature
Constant "C"
Time constant - radiative heating
Temperature After Equinox
Equilibrium Temperature
Operating Temperature

18) Finally the names of the data files are listed

B. MASS PROPELLENT BUDGET AND MASS SUMMARY

1) To start the program type "**MASSPRO**" at the dos prompt.

2) The program will ask "Is your spacecraft spin stabilized". If it is answer with a character 'Y' if not answer with a character 'N'

3) Program will state design chosen

a) "Spacecraft is Spin Stabilized" or

b) "Spacecraft is Three Axis Stabilized"

4) Program asks "What is your Spacecraft Mass"

a) Enter Spacecraft Mass including the Adaptor Mass in kilograms.

Example: "3000.0"

5) Program asks "Please enter the radius at Apogee"

Example "42353.0"

6) Program asks "Please enter the radius at Perigee"

Example "6565.0"

Program lists the values for:

r_P - Radius at Apogee

r_{ORBIT} - Radius of Transfer Orbit

V_{TP} - Velocity at perigee for the transfer orbit

V_{TA} - Velocity at apogee for the transfer orbit

V_S - Velocity of a Geosynchronous Orbit

5) Program asks "Please enter the launch inclination in degrees"

Example "25.2"

Program lists: $i_{DEGREES}$ - inclination in degrees

i_{RADIAN} - inclination in radian

Geosynchronous Orbit insertion angle

ΔV_{GT} - velocity to enter a geosynchronous equatorial orbit

6) Next program asks "Please enter the allowable inclination tolerance"

Example "0.2"

7) Next the program asks "Enter the spacecraft first year in orbit. Any year between 1991 and 2003" Example " 1994 "

8) Next program asks "Enter last year of spacecraft life " Example " 2001 "

9) Drift rates for the selected range of years is displayed and the program lists:

Average drift rate per year

T_{NS} - time between north south station keeping maneuvers

N_{NS} - number of north south station keeping maneuvers

ΔV_{NS} - delta velocity north south

10) Next program asks "Please enter the spacecraft operating longitude in degrees"

Example "312.3" then program lists:

$\lambda_{double\ dot}$ - longitudinal drift acceleration

T_{EW} - Time between east west station keeping maneuvers

ΔV_{EW} - delta velocity over spacecraft life for E-W station keeping

Next program states "the efficiency of the station keeping motors"

EFF_{NS} - Propulsion efficiency north south

EFF_{EW} - Propulsion efficiency east west

to change default values enter a '1' otherwise a '2'

11) Next program asks "Please enter the number of days allowed for station repositioning" Example "29.0" and then the program asks "Please enter the number of degrees to reposition.Example "125.9".

12) Program asks "What is your Spacecraft Mass"

a) Enter Spacecraft Mass including the Adaptor Mass in kilograms.

Example: "3000.0"

13) Next program asks "Please enter the I_{SP} for Apogee Injection

Example "285.0"

14) Next program asks "Please enter the efficiency of the motor for station repositioning"

Example "0.97"

15) Next program asks "Please enter the efficiency of the motor for satellite deorbit"

Example "0.98"

The program then lists the following values:

Pre AMF Fuel Mass

AMF Fuel Mass

Post AMF Fuel Mass

Mass change Post AMF

16) Next program asks "Please enter the I_{SP} for orbit maintenance" Example "285.0" Then the screen displays the following I_{SP}

ΔM_{NS}

ΔM_{EW}

ΔM_{SR}

$\Delta M_{ORBIT\ CONTROL}$

$M_{PRESSURANT}$

M_{MARGIN}

$\Delta M_{PROPELLENT}$

$M_{\text{STRUCTURE}}$

$M_{\text{SPACECRAFT BEGINNING OF LIFE}}$

$M_{\text{THERMAL CONTROL}}$

$M_{\text{ATTITUDE CONTROL}}$

$M_{\text{MECHANICAL SYSTEM}}$

$M_{\text{PROPELLENT}}$

$M_{\text{PROPELLENT PRESSURANT}}$

M_{DRYMASS}

$M_{\text{MASS MARGIN}}$

$M_{\text{COMMUNICATIONS PACKAGE}}$

17) Program asks "Enter the POWER requirements of the Spacecraft in watts.

Example: "2000.0"

18) Next the program asks you to chose a reference satellite from those listed

'1' Intelsat V

'2' Intelsat VI

'3' Intelsat VII

'4' or to inseıt your own values.

Screen displays values for:

$P_{\text{HOUSEKEEPING}}$

$P_{\text{BATTERY LOAD}}$

$P_{\text{SOLAR ARRAY LOAD}}$

Power Factor

$M_{\text{ELECTRICAL POWER SUBSYSTEM}}$

M_{PAYLOAD}

$M_{\text{TRACKING AND TELEMETRY}}$

and finally lists the data files for the design run.

C. ELECTRICAL POWER SYSTEM

1) To start the program type "**THERMAL**" at the dos prompt.

2) The program will ask "Is your spacecraft spin stabilized". If it is answer with a character 'Y' if not answer with a character 'N'

3) Program will state design chosen

a) "Spacecraft is Spin Stabilized" or

b) "Spacecraft is Three Axis Stabilized"

4) Program asks "What is your Spacecraft Mass"

a) Enter Spacecraft Mass including the Adaptor Mass in kilograms.

Example: "5000.0"

5) Program asks "Enter the POWER req rements of the Spacecraft in watts.

Example: "2000.0"

6) Next the program asks you to chose a reference satellite from those listed

'1' Intelsat V

'2' Intelsat VI

'3' Intelsat VII

'4' or to insert your own values.

7) The program lists the calculated housekeeping power and asks if you want to change this value for future calculations. If you want to change it enter a character 'Y' if not, depress a character 'N'.

Program then lists the values for Payload Power and Housekeeping Power

8) Program asks "Enter the spacecraft life in years" Example "10.0"

9) Program lists some default values for:

Minimum discharge bus voltage V_{DB}

Design satellite bus voltageV_{BUS}

Bypass diode voltage dropV_{DD}

EOL battery discharge voltageV_{MBC}

Satellite eclipse timeT_{EC}

Battery depth of dischargeDOD

Maximum battery discharge voltageV

Series connected diode voltage dropV

Number of series connected diodesN_{SCD}

Battery charger voltage dropV_{BC}

Charge discharge voltage dropV_{D}

To change any of the listed values enter a character 'Y' and then enter the number of the value you wish to change. After all desired changes have been made enter a character 'N'

and your final values will display on the screen

Example: if the user wants to change first the user enters a 'Y' then enters a "6" and the Program asks "Please enter the Depth of Discharge used for the batteries"

10) Program asks "Enter the number of electrical buses used in your satellite"
    Example "2"

11) Program asks "Please enter the Solar Cell Test Temperature" Example "28.0"

12) Program states some nice to know information.

13) Program states some calculated environmental design values and asks if you want to use them. To use calculated values enter a '1' to input your values enter a '2'.

14) Program displays some standard solar cell parameters and asks if the user wants to select one of these cells or enter their own values. User enters an integer number for the desired choice. Screen then displays selected or input parameters for user concurrence.

15) Screen display.

N_S - Number of solar cells in series

V_{DB} - Minimum discharge bus voltage

$CELL_{AH}$ - Battery cell ampere hours

P_{BUS} - Bus power

V_{MBC} - Maximum battery charge voltage

V_{BC} - Battery charger voltage drop

Boost Voltage

$I_{EQUINOX}$ - Equinox current

$I_{SOLSTICE}$ - Solstice current

P_{EC} - Power equinox charge

P_{SC} - Power solstice charge

$T_{RECHARGE}$ - Time to recharge the batteries

P_{SALEQU} - Solar array design load equinox

P_{SALSOL} - Solar array design load solstice

I_{MP} - Solar cell current at max power point EOL solstice

I_{MP} - Solar cell current at max power point EOL equinox

$I_{SOLSTICE}$ - Required current solstice per bus

$I_{EQUINOX}$ - Required current equinox per bus

V_{BUS} - Bus voltage

N_P - Number of solar cells in parallel for each bus

$V_{EOL\ SOLSTICE}$ - Solar cell voltage at EOL summer solstice

$V_{EOL\ EQUINOX}$ - Solar cell voltage at EOL autumnal equinox

N_S - Number of solar cells in series for each bus

I_{BUS} - Current per bus or wing

V_{BUS} - Voltage per bus or wing

P_{TOTAL} - Total power

N_C - Number of series cells for charge array solstice

N_{CSS} - Number of parallel cells for charge array solstice

N_{CEQ} - Number of parallel cells for charge array equinox

Finally the data file names appear for this design run.

SAMPLE DATA SHEET

SPACECRAFT MASS SUMMARY

Subsystem	Mass (kg)
Structure	278.4
Thermal	52.7
Propulsion	132.5
Attitude Contro	164.8
Electric Integration	76.1
Mechanical Integration	27.3
Mass Margin	162.3
Dry Spacecraft Mass	1622.8
Propellent Pressurant	384.3
Apogee Motor Expendable	1192.9
Spacecraft Mass at Seperation	3200.0
Communications	175.1
Antenna Reference Mass	309.0
Electric Power	351.6
Telemetry and Command	24.2

APPENDIX B

A. MASS PROPELLENT BUDGET AND SYSTEM MASS SUMMARY

```
-- Title          : Velocity Determination
-- Author          : David Lashbrook
-- Date           : 09 October 1991
-- Revised         : 05 May 1992
-- Compiler        : OPENADA EXT
-- Description    : This procedure determines the delta velocity for insertion
--                    into geosynchronous orbit.

with TEXT_IO, MATH_LIB,GETDATA, VIDEO;
use  TEXT_IO, MATH_LIB,GETDATA;

procedure MASSPRO is
  package FLOAT_INOUT is new FLOAT_IO(FLOAT);
  use     FLOAT_INOUT;
  package INTEGER_INOUT is new INTEGER_IO(INTEGER);
  use     INTEGER_INOUT;
  package BOOLEAN_INOUT is new ENUMERATION_IO(BOOLEAN);
  use     BOOLEAN_INOUT;

  INCLINATION_RADIANS,
  DELTA_VELOCITY_NORTH_SOUTH,
  DELTA_VELOCITY_EAST_WEST,
  DELTA_VELOCITY_STATION_REPOSITIONING,
  DELTA_VELOCITY,
  X             :  FLOAT;

  EFF_NS         : FLOAT := 0.91;
  EFF_EW          : FLOAT := 0.99;


  I                              :   INTEGER;


  MASS_BEFORE_APOGEE_BURN,
  SPACECRAFT_MASS_BEFORE_APOGEE_BURN,
  COMM_PACKAGE_MASS,
  SPACECRAFT_MASS                    :   FLOAT;

  DRUM_SPINNER                      :   BOOLEAN:=FALSE;

  OUTM                          :FILE_TYPE;
```

```
   procedure GET_DATA(X : out FLOAT) is
    begin
      loop
          begin
            NEW_LINE(2);
            SET_COL(10);
            PUT_LINE("Enter the value as a real number with a
decimal point");
            SET_COL(15);
            PUT_LINE("(Depress CTRL^C to exit the program.)");
            SET_COL(10);
            GET(X);
            SKIP_LINE;
            exit;
          exception
            when DATA_ERROR = >
           SKIP_LINE;
           NEW_LINE;
           SET_COL(10);
           PUT_LINE("Error.. You must enter the value as a
real");
           SET_COL(10);
           PUT_LINE("number with a decimal point.  ie 123.4");
           SET_COL(10);
           PUT_LINE("Try again.");
           NEW_LINE;
          end;
      end loop;
    end GET_DATA;

-- Reads an integer input from the keyboard

    procedure GET_INTEGER(I : out INTEGER) is
    begin
      loop
          begin
            NEW_LINE(2);
            SET_COL(10);
            PUT_LINE("Enter the value as an integer");
            PUT_LINE("(Depress CTRL^C to exit the program.)");
            SET_COL(10);
            GET(I);
            SKIP_LINE(1);
            exit;
          exception
            when DATA_ERROR = >
           SKIP_LINE;
           NEW_LINE;
           SET_COL(10);
           PUT_LINE("Error.. You must enter the value as a
INTEGER");
           SET_COL(10);
```

```
        PUT_LINE(" NO!  decimal point.  ie 123 ");
        SET_COL(10);
        PUT_LINE(" Please try again.");
        NEW_LINE;
       end;
    end loop;
  end GET_INTEGER;

 procedure GET_CHARACTER(CHAR : out CHARACTER) is
  begin
    loop
       begin
        NEW_LINE(2);
        SET_COL(10);
        PUT_LINE("Enter  'Y'  for YES or  ");
        NEW_LINE(1);
        SET_COL(10);
        PUT_LINE("        'N'  for NO");
        SET_COL(15);
        PUT_LINE("(Depress CTRL^C to exit the program.)");
        SET_COL(10);
        GET(CHAR);
        SKIP_LINE;
        exit;
       exception
        when DATA_ERROR =>
       SKIP_LINE;
       NEW_LINE;
       SET_COL(10);
       PUT_LINE("Error.. You must enter character");
       SET_COL(10);
       PUT_LINE("Try again.");
       NEW_LINE;
      end;
    end loop;
  end GET_CHARACTER;

  procedure DUAL_SPIN (DRUM_SPINNER : in out BOOLEAN) is
 y,
 Y,
 n,
 N,
 CHAR                          :    CHARACTER;

begin
  VIDEO.CLEAR_SCREEN;
  SET_COL(10);
  PUT_LINE("Is your spacecraft Spin Stabilized ");
  SET_COL(15);
  GET_CHARACTER(char);
  if CHAR = 'Y' or CHAR = 'y' then
    DRUM_SPINNER:=TRUE;
    if DRUM_SPINNER = TRUE then
    VIDEO.CLEAR_SCREEN;
    NEW_LINE(2);

PUT_LINE("*****************************************************
***********");
    NEW_LINE(2);
    SET_COL(10);
    PUT_LINE("Satellite is Spin Stabilized");
    NEW_LINE(2);

PUT_LINE("*****************************************************
***********");
    NEW_LINE(2);
    end if;
  else
    VIDEO.CLEAR_SCREEN;
    NEW_LINE(2);

PUT_LINE("*****************************************************
***********");
    NEW_LINE(2);
    SET_COL(10);
    PUT_LINE("Satellite is Three Axis Stabilized");
    NEW_LINE(2);

PUT_LINE("*****************************************************
***********");
    NEW_LINE(2);
  end if;
end DUAL_SPIN;

  procedure PRINT_HEADER is
   begin
    VIDEO.CLEAR_SCREEN;
    NEW_LINE(2);
    SET_COL(10);
    PUT_LINE("This program walks through a basic design of a");
    SET_COL(10);
    PUT_LINE("geosynchronous satellite.");
    NEW_LINE;
   end PRINT_HEADER;


procedure VELOCITY (INCLINATION_RADIANS   : in  out  FLOAT;
          DELTA_VELOCITY          : in  out  FLOAT) is



  RADIUS_EARTH              : FLOAT: =7378.0;             --
kilometers
  UE                     : constant FLOAT: =3.986E+05; --
```

```
km**3/seconds**2
 GEOSYNCHRONOUS_ORBIT_RADIUS : constant FLOAT:=4.2164E+04; -- km

 APOGEE_VELOCITY,
 PERIGEE_VELOCITY,
 SYNCHRONOUS_ORBIT_VELOCITY_CALCULATED,
 ANGULAR_VELOCITY,
 RADIUS_APOGEE,
 RADIUS_PERIGEE,
 APOGEE_ALTITUDE,
 PERIGEE_ALTITUDE,
 INCLINATION,
 ORBIT_RADIUS              : FLOAT:=0.0;           --
kilometers



 SYNCHRONOUS_ORBIT_VELOCITY  : constant FLOAT:=3.075;  --
kilometers/second

 W,
 X,
 Y,
 Z                      : FLOAT:=0.0;

 BETA,
 ALFA_DEGREES,
 ALFA                    : FLOAT:=0.0;          --
radians

 I,
 N                      : INTEGER;

 begin
   -- Read apogee radius from keyboard
   -- GET_APOGEE_RADIUS
     SET_COL(10);
     NEW_LINE(2);

PUT_LINE("*****************************************************
***********");
     NEW_LINE;
     PUT_LINE (" Please enter radius at apogee ");
     NEW_LINE(2);
     SET_COL(15);
     GET_DATA(RADIUS_APOGEE);
     VIDEO.CLEAR_SCREEN;
     NEW_LINE(2);
     SET_COL(10);
     PUT("Radius at apogee is  ");
     SET_COL(60);
```

```
    PUT(RADIUS_APOGEE, FORE => 6, AFT => 4, EXP => 0);
    PUT("  km");

  -- GET_PERIGEE_RADIUS is
    NEW_LINE(2);

PUT_LINE("**********************************************************
***********");
    NEW_LINE;
    SET_COL(10);
    PUT_LINE("Please enter radius at Perigee");
    GET_DATA(RADIUS_PERIGEE);
    NEW_LINE(2);
    SET_COL(10);
    VIDEO.CLEAR_SCREEN;
    PUT("Radius at perigee is  ");
    SET_COL(60);
    PUT(RADIUS_PERIGEE, FORE => 6, AFT => 4, EXP => 0);
    PUT("  km");

-- VELOCITIES

    ORBIT_RADIUS:=(RADIUS_APOGEE+RADIUS_PERIGEE)/2.0;
    NEW_LINE(2);
    SET_COL(10);
    PUT("Orbit radius is  ");
    SET_COL(60);
    PUT(ORBIT_RADIUS, FORE => 6, AFT => 2, EXP => 0);
    PUT("  km");


PERIGEE_VELOCITY:=SQRT((2.0*UE*RADIUS_APOGEE)/((RADIUS_APOGEE+RAD
IUS_PERIGEE)*RADIUS_PERIGEE));
    NEW_LINE(2);
    SET_COL(10);
    PUT("Perigee velocity is  ");
    SET_COL(60);
    PUT(PERIGEE_VELOCITY, FORE => 6, AFT => 4, EXP => 0);
    PUT("  km/sec");


APOGEE_VELOCITY:=PERIGEE_VELOCITY*(RADIUS_PERIGEE/RADIUS_APOGEE);
    NEW_LINE(2);
    SET_COL(10);
    PUT("Apogee velocity is  ");
    SET_COL(60);
    PUT(APOGEE_VELOCITY, FORE => 6, AFT => 4, EXP => 0);
    PUT("  km/sec");

--    ANGULAR_VELOCITY:= RADIUS_APOGEE*APOGEE_VELOCITY;
--    NEW_LINE(2);
--    SET_COL(10);
--    PUT("Angular (h) velocity is  ");
   SET_COL(60);
--    PUT(ANGULAR_VELOCITY, FORE => 6, AFT => 4, EXP => 0);

    SYNCHRONOUS_ORBIT_VELOCITY_CALCULATED:= SQRT
               (UE/GEOSYNCHRONOUS_ORBIT_RADIUS);
    NEW_LINE(2);
    SET_COL(10);
    PUT("Geosynchronous orbit velocity is  ");
    SET_COL(60);
    PUT(SYNCHRONOUS_ORBIT_VELOCITY_CALCULATED,FORE =>6,AFT
=>4,EXP=>0);
    PUT("  km/sec");NEW_LINE(2);




-- FIND_ANGLES
   NEW_LINE(2);

PUT_LINE("*********************************************************
***********");
   NEW_LINE(2);
   SET_COL(10);
   PUT_LINE("Please enter the inclination the launch");
   SET_COL(10);
   PUT_LINE("vehicle will insert the spacecraft.");
   NEW_LINE(2);
   GET_DATA(INCLINATION);
   INCLINATION_RADIANS:=INCLINATION*PI/180.0;
   VIDEO.CLEAR_SCREEN;
   NEW_LINE(2);
   SET_COL(10);
   PUT("Inclination in degrees is  ");
   SET_COL(60);
   PUT(INCLINATION, FORE => 6, AFT => 2, EXP => 0);
   NEW_LINE(2);
   SET_COL(10);
   PUT("Inclination in radians is  ");
   SET_COL(60);
   PUT(INCLINATION_RADIANS, FORE => 6, AFT => 5, EXP => 0);

    ALFA:=ATAN((APOGEE_VELOCITY*SIN(INCLINATION_RADIANS))

/(SYNCHRONOUS_ORBIT_VELOCITY-APOGEE_VELOCITY*COS(INCLINATION_RADI
ANS)));

     ALFA_DEGREES:=ABS(ALFA*180.0/PI);
     NEW_LINE(2);
     SET_COL(10);
     PUT("Insertion angle in degrees (ALFA) is  ");
     SET_COL(60);
     PUT(ALFA_DEGREES, FORE => 6, AFT => 2, EXP => 0);

   -- BETA:=ABS(180.0-(INCLINATION+ALFA_DEGREES));
   -- NEW_LINE(2);
   -- SET_COL(10);
   -- PUT("Triangular angle (BETA) is  ");
   -- PUT(BETA, FORE => 6, AFT => 2, EXP => 0);



  -- DELTA_VEL


DELTA_VELOCITY:=SQRT(((APOGEE_VELOCITY*SIN(INCLINATION_RADIANS))*
*2)
     +((SYNCHRONOUS_ORBIT_VELOCITY
     -APOGEE_VELOCITY*COS(INCLINATION_RADIANS))**2));
     NEW_LINE(2);
     SET_COL(10);
     PUT("Delta velocity for insertion into geo is  ");
     SET_COL(60);
     PUT(DELTA_VELOCITY, FORE => 6, AFT => 4, EXP => 0);
     PUT("  km/sec");
     NEW_LINE(2);

PUT_LINE("*******************************************************
***********");
     NEW_LINE(2);

end VELOCITY;

procedure STATION_KEEPING_REPOSITIONING

     (DELTA_VELOCITY_NORTH_SOUTH          : in out FLOAT;
      DELTA_VELOCITY_EAST_WEST            : in out FLOAT;
      DELTA_VELOCITY_STATION_REPOSITIONING  : in out FLOAT;
      EFF_NS                              : in out FLOAT;
      EFF_EW                              : in out FLOAT) is

    AVERAGE_INCLINATION_DRIFT_YEAR,
    AVERAGE_DRIFT_PER_YEAR,
    AVERAGE_DRIFT_YEAR,
    ADPY,
    LAMDA_DOT_DOT,
    LAMDA,
    DELTA_LAMDA,

    OPERATING_LONGITUDE,
    LONGITUDINAL_DRIFT_ACCELERATION,
    TOLERANCE,
    TOLERANCE_RADIANS,
    INCLINATION_DRIFT_YEAR   : FLOAT :=0.0;
    NORTH_SOUTH_MANEUVERS,
    NORTH_SOUTH_MANEUVERS_INTEGER,
    TIME_NS,
    FUEL_MASS_NS,
    TIME_EW,
    ISP_FUEL,
    EAST_WEST_MANEUVERS,
    FUEL_MASS_EW,
    DELTA_VELOCITY_TOTAL, -- delta velocity required for both
east west
              -- and north south station keeping

    --EFF_NS,    -- thruster efficiency north south station
keeping
    --EFF_EW,    -- thruster efficiency east west station
keeping

    X,
    EFF_DOR,  -- thruster efficiency de-orbit
    EFF_SR,   -- thruster efficiency station repositioning

    STABLE_LONGITUDE,
    DAYS_TO_REPOSITION,
    DEGREES_TO_REPOSITION,
    DELTA_FUEL_STATION_REPOSITIONING,
    EFF_STATION_REPOSITION    :        FLOAT;

    WORST_LONGITUDINAL_DRIFT_ACCELERATION : constant FLOAT :=
-0.00168;
    STABLE_LONGITUDE_EAST : constant := 75.0;
    STABLE_LONGITUDE_WEST : constant := 255.0;
    GRAVITY               : constant :=9.81;    --m/s

    FIRST_YEAR : INTEGER        := 1991;
    LAST_YEAR  : INTEGER        := 2003;

    N,
    I,
    CHOICE,
    SPACECRAFT_LIFE :                         INTEGER;

    type DRIFT_PER_YEAR is array (FIRST_YEAR..LAST_YEAR) of
FLOAT;
    DRIFT: DRIFT_PER_YEAR
    :=(0.897,0.867,0.834,0.802,0.775,0.756,0.748,0.752,
    0.767,0.792,0.823,0.856,0.888);

-- 1991 through 2003

begin
  NEW_LINE(2);
```

```
PUT_LINE("*******************************************************************  ');
  NEW_LINE(2);
  SET_COL(10);
  PUT_LINE("Enter the tolerance of spacecraft LATITUDE as real
number.");
  GET_DATA(TOLERANCE);
  VIDEO.CLEAR_SCREEN;
  PUT("Tolerance is ")·
  set_col(60);
  PUT(TOLERANCE,FORE=>6,AFT=>2,EXP=>0);PUT("  deg");
  TOLERANCE_RADIANS:=TOLERANCE*PI/180.0;
  SET_COL(10);


<<YEAR>>

  NEW_LINE(2);

PUT_LINE("*******************************************************************");
  NEW_LINE;
  PUT_LINE("Please enter the beginning year as an integer");
  SET_COL(5);
  GET_INTEGER (FIRST_YEAR);
  VIDEO.CLEAR_SCREEN;
  if FIRST_YEAR < 1991 then
    VIDEO.CLEAR_SCREEN;
    PUT_LINE("ERROR ..... First Year is lower than 1991");
    PUT_LINE("Please Try Again.");
    NEW_LINE(3);
    goto year;
  end if;
  SET_COL(10);
  NEW_LINE(2);
  PUT("First year of lifetime is");
  NEW_LINE(2);

PUT_LINE("*******************************************************************");
  NEW_LINE(2);
  SET_COL(60);
  PUT(FIRST_YEAR,WIDTH=>5);
  NEW_LINE(2);
  SET_COL(5);

 <<YEARA>>

  PUT_LINE("Please enter the ending year as an integer");
  SET_COL(5);
  GET_INTEGER (LAST_YEAR);
  VIDEO.CLEAR_SCREEN;
  NEW_LINE(2);
```

```
PUT_LINE("**********************************************************
***********");
  NEW_LINE;
  if LAST_YEAR < FIRST_YEAR  then
    VIDEO.CLEAR_SCREEN;
    PUT_LINE("ERROR ..... :Last Year is lower than First
Year");
    PUT_LINE("Please Try Again.");
    NEW_LINE(3);
    goto YEAR;
  elsif LAST_YEAR > 2003  then
    VIDEO.CLEAR_SCREEN;
    PUT_LINE("ERROR ..... Last Year is greater than 2003");
    PUT_LINE("Please Try Again.");
    NEW_LINE(3);
    goto YEARA;
  end if;
  SET_COL(5);
  PUT("Last year of expected lifetime is");
  SET_COL(60);
  PUT(LAST_YEAR,WIDTH=>5);
  NEW_LINE(2);

PUT_LINE("**********************************************************
***********");
  NEW_LINE(2);

  SPACECRAFT_LIFE:=LAST_YEAR-FIRST_YEAR+1; -- Total years
includes ending year
  NEW_LINE(2);
  SET_COL(5);
  PUT("Spacecraft life is ");
  PUT(SPACECRAFT_LIFE, width=>3);
  PUT(" years");

  PUT("   (launch year counts as one year)");
  NEW_LINE(2);

PUT_LINE("**********************************************************
***********");
  NEW_LINE(2);

  SET_COL(10);PUT("TO CONTINUE ENTER ANY INTEGER");
GET_INTEGER(I);



  VIDEO.CLEAR_SCREEN;

PUT_LINE("**********************************************************
***********");
```

```
  NEW_LINE;
  SET_COL(5);
  PUT("Drift per year ");
  while FIRST_YEAR <= LAST_YEAR loop

INCLINATION_DRIFT_YEAR:=INCLINATION_DRIFT_YEAR+DRIFT(FIRST_YEAR);


    SET_COL(5);
    PUT(FIRST_YEAR,WIDTH=> 5);
    SET_COL(20);
    PUT(DRIFT(FIRST_YEAR), FORE=> 1, AFT=> 4, EXP=>0);
    FIRST_YEAR:=FIRST_YEAR+1;

  end loop;
  NEW_LINE(2);

PUT_LINE("*********************************************************
***********");
  NEW_LINE(2);

  SET_COL(10);PUT("TO CONTINUE ENTER ANY INTEGER");
GET_INTEGER(I);


AVERAGE_DRIFT_PER_YEAR:=INCLINATION_DRIFT_YEAR/FLOAT(SPACECRAFT_L
IFE);
  NEW_LINE(2);
  ADPY:=AVERAGE_DRIFT_PER_YEAR;
  SET_COL(5);
  NEW_LINE(2);

PUT_LINE("*********************************************************
***********");
  NEW_LINE(2);

  VIDEO.CLEAR_SCREEN;
  SET_COL(5);
  PUT(" Average drift rate per year is ");
  SET_COL(60);
  PUT(ADPY, FORE => 4, AFT => 4, EXP => 0);
  PUT("  degrees");

  TIME_NS:=(2.0*TOLERANCE/ADPY)*365.25;
  NEW_LINE(2);
  SET_COL(5);
  PUT(" Time spent in north south station keeping is ");
  SET_COL(60);
  PUT(TIME_NS, FORE => 4, AFT => 4, EXP => 0);
  PUT("  days");
```

```
NORTH_SOUTH_MANEUVERS:=(FLOAT(SPACECRAFT_LIFE)*ADPY)/(2.0*TOLERAN
CE);
  NEW_LINE(2);
  SET_COL(5);
  PUT(" Number of north south maneuvers is (REAL)");
  SET_COL(60);
  PUT(NORTH_SOUTH_MANEUVERS, FORE => 4, AFT => 4, EXP => 0);

  N:=INTEGER(NORTH_SOUTH_MANEUVERS);
  NORTH_SOUTH_MANEUVERS_INTEGER:=FLOAT(N);
  --NEW_LINE(2);
  --SET_COL(5);
  --PUT(" N");
  --PUT(NORTH_SOUTH_MANEUVERS, FORE=>4,AFT=>4,EXP => 0);


  if NORTH_SOUTH_MANEUVERS_INTEGER < NORTH_SOUTH_MANEUVERS then

NORTH_SOUTH_MANEUVERS_INTEGER:=NORTH_SOUTH_MANEUVERS_INTEGER+1.0;
  end if;

  NORTH_SOUTH_MANEUVERS:=NORTH_SOUTH_MANEUVERS_INTEGER;
  NEW_LINE(1);
  SET_COL(5);
  PUT(" Number of north south maneuvers is (ROUND UP)");
  SET_COL(60);
  PUT(NORTH_SOUTH_MANEUVERS, FORE => 4, AFT => 4, EXP => 0);

  DELTA_VELOCITY_NORTH_SOUTH:=
  NORTH_SOUTH_MANEUVERS*3.148*SIN(TOLERANCE_RADIANS)*1000.0;
  NEW_LINE(2);
  SET_COL(5);
  PUT(" Delta velocity north south is");
  SET_COL(60);
  PUT(DELTA_VELOCITY_NORTH_SOUTH, FORE => 4, AFT => 4, EXP =>
0);
  PUT("  m/sec");

  NEW_LINE(2);

PUT_LINE("*****************************************************
***********");
  NEW_LINE(2);


  SET_COL(10);PUT("TO CONTINUE ENTER ANY INTEGER");
GET_INTEGER(I);



  -- East West station keeping
  VIDEO.CLEAR_SCREEN;
  NEW_LINE(2);

PUT_LINE("********************************************************
***********");
  NEW_LINE(2);

  SET_COL(5);
  PUT_LINE("Enter the spacecraft OPERATING LONGITUDE");
  NEW_LINE(2);
  SET_COL(5);
  GET_DATA(OPERATING_LONGITUDE);
  VIDEO.CLEAR_SCREEN;
  NEW_LINE(2);
  SET_COL(10);
  PUT("Operating Longitude is ");
  PUT(OPERATING_LONGITUDE, FORE => 6, AFT => 2, EXP => 0);
  PUT("  degrees longitude");
  NEW_LINE(2);

PUT_LINE("********************************************************
***********");
  NEW_LINE(2);
  SET_COL(5);

  if OPERATING_LONGITUDE > 345.0000
    and OPERATING_LONGITUDE <=360.0 then
    DELTA_LAMDA:=SIN(2.0*PI/180.0*
    (360.0-OPERATING_LONGITUDE+STABLE_LONGITUDE_EAST));
    STABLE_LONGITUDE:=STABLE_LONGITUDE_EAST;

  elsif OPERATING_LONGITUDE >= 0.0
    and OPERATING_LONGITUDE <165.0 and OPERATING_LONGITUDE /=
75.0 then
    DELTA_LAMDA:=SIN(2.0*PI/180.0*
    (OPERATING_LONGITUDE-STABLE_LONGITUDE_EAST));
    STABLE_LONGITUDE:=STABLE_LONGITUDE_EAST;

  elsif OPERATING_LONGITUDE > 165.0
    and OPERATING_LONGITUDE <345.0 and OPERATING_LONGITUDE /=
255.0 then
    DELTA_LAMDA:=SIN(2.0*PI/180.0*
    (OPERATING_LONGITUDE-STABLE_LONGITUDE_WEST));
    STABLE_LONGITUDE:=STABLE_LONGITUDE_WEST;

  elsif OPERATING_LONGITUDE = 165.0 or OPERATING_LONGITUDE =
345.0  then
    DELTA_LAMDA:= WORST_LONGITUDINAL_DRIFT_ACCELERATION;
    SET_COL(5);
    PUT("Delta Lamda is ");
    PUT(DELTA_LAMDA,FORE=>4,AFT=>2,EXP=>0);
    PUT("  degrees");
  elsif OPERATING_LONGITUDE = 255.0 or OPERATING_LONGITUDE =
75.0 then
      SET_COL(5);
      PUT(" Stable longitude so time between east-west is");
      SET_COL(5);PUT("essentially infinite");
      TIME_EW:=0.0;
      NEW_LINE(2);
      SET_COL(5);
      PUT(" Time between east west keeping is ");
      PUT(TIME_EW, FORE => 4, AFT => 4, EXP => 0);
      NEW_LINE(2);
      SET_COL(5);
      LAMDA_DOT_DOT:=0.0;
      SET_COL(5);
      PUT("Delta Lamda DOUBLE DOT is ");
      PUT(LAMDA_DOT_DOT,FORE=>1,AFT=>7,EXP=>0);
      PUT("  degrees/day^2");

  end if;

  if OPERATING_LONGITUDE /= 75.0  then
      if OPERATING_LONGITUDE /= 255.0 then

LAMDA_DOT_DOT:=ABS((WORST_LONGITUDINAL_DRIFT_ACCELERATION)*(DELTA
_LAMDA));
      NEW_LINE(2);
      SET_COL(5);
      PUT("Delta Lamda DOUBLE DOT is ");
      SET_COL(50);
      PUT(LAMDA_DOT_DOT,FORE=>1,AFT=>7,EXP=>0);
      PUT("  degrees/day^2");


   -- Average time interval between east-west station keeping
days

      TIME_EW:= 4.0*SQRT((TOLERANCE/LAMDA_DOT_DOT));
      NEW_LINE(2);
      SET_COL(5);
      PUT(" Time between east west keeping is ");
      SET_COL(50);
      PUT(TIME_EW, FORE => 4, AFT => 4, EXP => 0);
      PUT("  days");
      end if;
   end if;




   DELTA_VELOCITY_EAST_WEST:=ABS(1./4*SIN(2.0*PI/180.0
*(OPERATING_LONGITUDE-STABLE_LONGITUDE))*FLOAT(SPACECRAFT_LIFE));

  NEW_LINE(2);
  SET_COL(5);
  PUT(" Delta velocity east west is");
  SET_COL(50);
  PUT(DELTA_VELOCITY_EAST_WEST, FORE => 4, AFT => 4, EXP => 0);
  PUT("  m/sec");


  -- Total delta velocity required
  NEW_LINE(2);
  SET_COL(5);
  PUT_LINE("The default propulsion efficiency in the N-S
direction is  0.91");
  SET_COL(5);
  SET_COL(5);
  PUT_LINE("The default propulsion efficiency in the E-W
direction is  0.99");
  NEW_LINE(2);

  SET_COL(5);
  PUT_LINE("enter an integer  '1'   to CHANGE default values");
  NEW_LINE(2);
  SET_COL(5);
  PUT_LINE("To accept default values enter integer  2 ");
  GET_INTEGER(CHOICE);

  case CHOICE is
    when 1=>
    VIDEO.CLEAR_SCREEN;
    SET_COL(5);
    PUT_LINE("Please enter desired N-S propulsion efficiency");
    SET_COL(5);
    GET_DATA(EFF_NS);
    NEW_LINE(2);
    SET_COL(5);
    PUT_LINE("Now enter desired E-W propulsion efficiency");
    SET_COL(5);
    GET_DATA(EFF_EW);
    VIDEO.CLEAR_SCREEN;
    when OTHERS =>
    VIDEO.CLEAR_SCREEN;
    NEW_LINE(2);
    SET_COL(5);
    PUT_LINE ("Understand no changes desired");
    NEW_LINE(3);
  end case;

  DELTA_VELOCITY_TOTAL:=(DELTA_VELOCITY_NORTH_SOUTH/EFF_NS)
    +(DELTA_VELOCITY_EAST_WEST/EFF_EW);
  NEW_LINE(2);
  SET_COL(10);
  PUT("Delta velocity total is ");
  PUT(DELTA_VELOCITY_TOTAL, FORE => 6, AFT => 2, EXP => 0);
  PUT ("  km/sec");
  NEW_LINE(2);

PUT_LINE("*********************************************************
***********");
  NEW_LINE(2);

    SET_COL(10);PUT("TO CONTINUE ENTER ANY INTEGER");
GET_INTEGER(I);
    VIDEO.CLEAR_SCREEN;


  -- STATION REPOSITIONING
  SET_COL(5);
  PUT_LINE("Enter the numbers of days to reposition");
  SET_COL(5);
  GET_DATA(DAYS_TO_REPOSITION);
  VIDEO.CLEAR_SCREEN;
  NEW_LINE(2);
  SET_COL(10);
  PUT("Number of Days to Reposition is ");
  SET_COL(50);
  PUT(DAYS_TO_REPOSITION, FORE => 6, AFT => 2, EXP => 0);
  PUT ("  days");
  NEW_LINE(2);

PUT_LINE("*********************************************************
***********");
  NEW_LINE(2);
  SET_COL(5);

  -- Get how many degrees to reposition satellite

  SET_COL(5);
  PUT_LINE("Enter the number of degrees to reposition");
  SET_COL(5);
  PUT_LINE("as real number.");
  GET_DATA(DEGREES_TO_REPOSITION);
  VIDEO.CLEAR_SCREEN;
  NEW_LINE(2);

PUT_LINE("*********************************************************
***********");
  NEW_LINE(2);
  SET_COL(10);
  PUT("Number of Degrees to Reposition is ");
  SET_COL(50);
  PUT(DEGREES_TO_REPOSITION, FORE => 6, AFT => 2, EXP => 0);
  PUT("  degrees");
  -- Delta velocity to reposition satellite

  DELTA_VELOCITY_STATION_REPOSITIONING:=5.66*
    DEGREES_TO_REPOSITION/DAYS_TO_REPOSITION;
  NEW_LINE(2);
  SET_COL(10);
  PUT("Delta velocity for station repositioning is");
  SET_COL(50);
  PUT(DELTA_VELOCITY_STATION_REPOSITIONING, FORE => 6, AFT => 2,
EXP => 0);
  PUT("  m/sec");

  NEW_LINE(2);

PUT_LINE("*******************************************************
***********");
  NEW_LINE(2);


end STATION_KEEPING_REPOSITIONING;

procedure MASS

     (DELTA_VELOCITY_NORTH_SOUTH              : in         FLOAT;
     DELTA_VELOCITY_EAST_WEST              : in         FLOAT;
     DELTA_VELOCITY_STATION_REPOSITIONING  : in         FLOAT;
     DELTA_VELOCITY                        : in         FLOAT;
     EFF_NS                     : in         FLOAT;
     EFF_EW                      : in         FLOAT;
     SPACECRAFT_MASS_BEFORE_APOGEE_BURN   : in out    FLOAT;
     COMM_PACKAGE_MASS                    : in out     FLOAT)
is


  gravity : constant FLOAT        := 9.81;            -- m/s
  Y       : constant FLOAT        := 7.0;             --
constant
  ADAPTOR : constant FLOAT        := 45.0;
  ISP_ORBIT : FLOAT               := 278.0;           -- sec
  MASS_REFERENCE : FLOAT          := 1900.0;          --
kilograms
  SPACECRAFT : FLOAT;
  PRE_AMF_REFERENCE : FLOAT       := 7.0;             -- kg
  AMF_REFERENCE : FLOAT           := 861.0;           -- kg
  POST_AMF_REFERENCE: FLOAT       := 29.9;            -- kg
  MASS_CHANGE_POST_AMF:FLOAT      := 0.0;
  PRESSURANT_REF: FLOAT           := 5.0;             -- kg
  MARGIN_REF: FLOAT               := 2.0;             -- kg
  ON_ORBIT_CONTROL_REF :FLOAT     := 118.0;           -- kg
  ATTITUDE_CONTROL_REF :FLOAT     := 12.3;            -- kg
  ISP_AMF : FLOAT                 := 300.0;           -- sec
  DEORBIT_REFERENCE : FLOAT       := 5.2;             -- kg
```

```
PRESSURANT_REFERENCE: FLOAT     := 5.0;              -- kg
MARGIN_REFERENCE : FLOAT        := 2.0;              -- kg
DELTA_VELOCITY_DEORBIT:FLOAT    := 7.0;


SCALE_FACTOR : constant FLOAT   := 1.9;          --
FACTOR : constant FLOAT         := 1.05;        --
MASS_RATIO : FLOAT;

PRE_AMF,
AMF,
POST_AMF,
BOL_MASS,
ON_ORBIT_CONTROL,
ATTITUDE_CONTROL,
DEORBIT,
PRESSURANT,
MARGIN,


SPACECRAFT_DRY_MASS,
SPACECRAFT_BOL_MASS,

APOGEE_MOTOR_IMPULSE,
ORBIT_IMPULSE,
MASS_SPACECRAFT,
SPACECRAFT_MASS,
MASS_POST_AMF,
MASS_EW_STATION_KEEPING,
MASS_NS_STATION_KEEPING,
MASS_STATION_REPOSITIONING,
MASS_DEORBIT,

MASS_DRY,
MASS_INITIAL,
MASS_MARGIN,
MASS_PROPELLENT_PRESSURANT,

PROPELLENT_MARGIN,
PROPELLENT_MASS,
PROPELLENT_EXPENDITURE,
PROPELLENT_PRESSURANT_MASS,
PROPELLENT_MASS_CHANGE,
PROPELLENT_EXPENDITURE,
BYPROPELLENT_MASS,

STRUCTURAL_MASS,
THERMAL_CONTROL_MASS,
ELECTRICAL_SYSTEM_MASS,
MECHANICAL_SYSTEM_MASS,
DRY_MASS,
PAYLOAD_POWER,
  HOUSEKEEPING_POWER,

  BATTERY_LOAD,
  SOLAR_ARRAY_LOAD,
  ELECTRICAL_POWER_MASS,
  EFF_STATION_REPOSITIONING,
  EFF_DEORBIT,
  PROPELLENT_MARGIN,
  X,
  Z,
  ON_ORBIT_CONTROL,
  TRACKING_TELEMETRY              : FLOAT ;
  N                       : INTEGER ;
  --DELTA_VELOCITY : constant FLOAT := 1.52 ;

  OUTF            :FILE_TYPE;




begin



 SET_COL(10);
 PUT_LINE("Enter the mass of the spacecraft in kilograms");
 SET_COL(10);
 GET_DATA(SPACECRAFT_MASS_BEFORE_APOGEE_BURN);
 NEW_LINE(2);
 SET_COL(15);
 VIDEO.CLEAR_SCREEN;
 NEW_LINE(2);
 PUT("Spacecraft mass before apogee motor burn is  ");
 PUT(SPACECRAFT_MASS_BEFORE_APOGEE_BURN, FORE => 6, AFT => 2. EXP
=> 0);
 PUT("  kgs");
 NEW_LINE(2);

PUT_LINE("*******************************************************
***********");
 NEW_LINE(2);
 SET_COL(10);
 PUT("What is the specific impulse for apogee injection");
 SET_COL(15);
 get_data(APOGEE_MOTOR_IMPULSE);
 VIDEO.CLEAR_SCREEN;

PUT_LINE("*******************************************************
***********");
 NEW_LINE;
 PUT("Specific Impulse of Apogee injection is  ");
 PUT(APOGEE_MOTOR_IMPULSE, FORE =>4 , AFT => 1, EXP => 0);
 PUT(" sec");

  NEW_LINE(2);

PUT_LINE("*****************************************************
***********");
  NEW_LINE(2);


--SPACECRAFT_MASS_BEFORE_APOGEE_BURN:=MASS_BEFORE_APOGEE_BURN-ADA
PTOR;
  SET_COL(10);
  PUT_LINE("  This mass budget uses modified INTELSAT V");
  SET_COL(10);
  PUT_LINE ("data for a premliminary estimation purposes");
  SET_COL(10);
  PUT_LINE("the reference data to a different satellite
simply");
  SET_COL(10);
  PUT_LINE ("changing the values labelled reference in the");
  SET_COL(10);
  PUT_LINE ("declaration statements");

  NEW_LINE(3);
  SET_COL(5);
  PUT("Please enter the value for efficiency of station
repositioning");

  SET_COL(15);
  get_data(EFF_STATION_REPOSITIONING);
  VIDEO.CLEAR_SCREEN;
  NEW_LINE(2);
  SET_COL(10);
  PUT("Efficiency of STATION REPOSITIONING is  ");
  SET_COL(50);
  PUT(EFF_STATION_REPOSITIONING, FORE =>1 , AFT => 3, EXP => 0);
  NEW_LINE(2);

PUT_LINE("*****************************************************
***********");
  NEW_LINE(2);
  SET_COL(5);
  PUT("Please enter the value for efficiency of satellite
deorbit");
  SET_COL(15);
  get_data(EFF_DEORBIT);
  VIDEO.CLEAR_SCREEN;
  NEW_LINE(2);
  SET_COL(15);
  PUT("Efficiency of deorbit is  ");
  PUT(EFF_DEORBIT, FORE =>1 , AFT => 3, EXP => 0);
 NEW_LINE(2);

PUT_LINE("*********************************************************
***********");
 NEW_LINE(2);
  MASS_RATIO:=SPACECRAFT_MASS_BEFORE_APOGEE_BURN/MASS_REFERENCE;

  NEW_LINE(2);
  SET_COL(10);
  PUT("Mass Ratio is  ");
  SET_COL(50);
  PUT(MASS_RATIO , FORE => 6, AFT => 2, EXP => 0);

  SET_COL(10); PUT("TO CONTINUE ENTER ANY INTEGER");
GET_INTEGER(I);
  VIDEO.CLEAR_SCREEN;

 PRE_AMF:=MASS_RATIO*PRE_AMF_REFERENCE;
  NEW_LINE(2);
  SET_COL(10);
  PUT("Pre amf is ");
  SET_COL(50);
  PUT(PRE_AMF , FORE => 6, AFT => 2, EXP => 0);
  PUT("  kgs");

  AMF:=SPACECRAFT_MASS_BEFORE_APOGEE_BURN

*(1.0-EXP((-DELTA_VELOCITY*1000.0)/(APOGEE_MOTOR_IMPULSE*GRAVITY)
));
  NEW_LINE(2);
  SET_COL(10);
  PUT("AMF is  ");
  SET_COL(50);
  PUT(AMF , FORE => 6, AFT => 2, EXP => 0);
  PUT("  kgs");


  POST_AMF:=SPACECRAFT_MASS_BEFORE_APOGEE_BURN-AMF-PRE_AMF;




  NEW_LINE(2);
  SET_COL(10);
  PUT("Post AMF is  ");
  SET_COL(50);
  PUT(POST_AMF , FORE => 6, AFT => 2, EXP => 0);

  PUT("  kgs");

  MASS_CHANGE_POST_AMF:=MASS_RATIO*POST_AMF_REFERENCE;
  NEW_LINE(2);
  SET_COL(10);
  PUT("Mass Change POST AMF is  ");
  SET_COL(50);
  PUT(MASS_CHANGE_POST_AMF, FORE => 6, AFT => 2, EXP => 0);
  PUT(" kgs");

  NEW_LINE(2);

PUT_LINE("*******************************************************
***********");
  NEW_LINE(2);


  SET_COL(10);
 PUT("What is the specific impulse for ORBIT maintainance");
 SET_COL(15);
 get_data(ORBIT_IMPULSE);
 VIDEO.CLEAR_SCREEN;
 PUT("Specific Impulse of orbit maintainance is  ");
 SET_COL(60);
 PUT(ORBIT_IMPULSE, FORE =>4 , AFT => 1, EXP => 0);
 PUT(" sec");
 NEW_LINE(1);

PUT_LINE("*******************************************************
***********");

  MASS_NS_STATION_KEEPING:=POST_AMF*(1.0-

EXP((-DELTA_VELOCITY_NORTH_SOUTH)/(ORBIT_IMPULSE*GRAVITY*EFF_NS))
);
  NEW_LINE(1);
  SET_COL(10);
  PUT("Change in mass for north south station keeping is");
  SET_COL(60);
  PUT(MASS_NS_STATION_KEEPING, FORE => 6, AFT => 2, EXP => 0);
  PUT(" kgs");


MASS_EW_STATION_KEEPING:=(POST_AMF-MASS_NS_STATION_KEEPING)*(1.0-
              EXP((-DELTA_VELOCITY_EAST_WEST)
              /(ORBIT_IMPULSE*GRAVITY*EFF_EW)));
  NEW_LINE(2);
  SET_COL(10);
  PUT("Change in mass for east west station keeping is");
  SET_COL(60);
  PUT(MASS_EW_STATION_KEEPING, FORE => 6, AFT => 2, EXP => 0);
  PUT(" kgs");

  MASS_STATION_REPOSITIONING:=(POST_AMF-MASS_NS_STATION_KEEPING
          -MASS_EW_STATION_KEEPING)*(1.0-
          EXP((-DELTA_VELOCITY_STATION_REPOSITIONING)
/(ORBIT_IMPULSE*GRAVITY*EFF_STATION_REPOSITIONING)));
  NEW_LINE(2);
  SET_COL(10);
  PUT("Change in mass for station repositioning is");
  SET_COL(60);
  PUT(MASS_STATION_REPOSITIONING, FORE => 6, AFT => 2, EXP =>
0);
  PUT("  kgs");


--**************************************************************

  ON_ORBIT_CONTROL:=POST_AMF*(1.0
  -EXP((-DELTA_VELOCITY_DEORBIT)/(ORBIT_IMPULSE*GRAVITY)));
  NEW_LINE(2);
  SET_COL(10);
  PUT("On orbit control is  ");
  SET_COL(60);
  PUT(ON_ORBIT_CONTROL , FORE => 6, AFT => 2, EXP => 0);
  PUT("  kgs");

--**************************************************************


  MASS_DEORBIT:=(POST_AMF-MASS_NS_STATION_KEEPING
  -MASS_EW_STATION_KEEPING-MASS_STATION_REPOSITIONING)*(1.0-

EXP((-DELTA_VELOCITY_DEORBIT)/(ORBIT_IMPULSE*GRAVITY*EFF_DEORBIT)
));
  NEW_LINE(2);
  SET_COL(10);
  PUT("Change in mass for de-orbit is");
  SET_COL(60);
  PUT(MASS_DEORBIT, FORE => 6, AFT => 2, EXP => 0);
  PUT("  kgs");


  PRESSURANT:=MASS_RATIO*PRESSURANT_REF;
  NEW_LINE(2);
  SET_COL(10);
  PUT("Pressurant is  ");
  SET_COL(60);
  PUT(PRESSURANT , FORE => 6, AFT => 2, EXP => 0);
  PUT("  kgs");

PROPELLENT_MARGIN:=(MASS_NS_STATION_KEEPING+MASS_EW_STATION_KEEPI
NG
           + MASS_STATION_REPOSITIONING+MASS_DEORBIT
           + PRE_AMF+MASS_CHANGE_POST_AMF+PRESSURANT)*0.02;
  NEW_LINE(2);
  SET_COL(10);
  PUT("Propellent margin is (2% safety margin) ");
  SET_COL(60);
  PUT(PROPELLENT_MARGIN, FORE => 6, AFT => 2, EXP => 0);
  PUT(" kgs");

-- Total mass change due to propellent expenditure
  PROPELLENT_EXPENDITURE:=(PROPELLENT_MARGIN*51.0)+AMF;
  NEW_LINE(2);
  SET_COL(10);
  PUT("Propellent Expenditure is");
  SET_COL(60);
  PUT(PROPELLENT_EXPENDITURE, FORE => 6, AFT => 2, EXP => 0);
  PUT(" kgs");

  SET_COL(10); PUT("TO CONTINUE ENTER ANY INTEGER");
GET_INTEGER(I);
  video.clear_screen;

STRUCTURAL_MASS:=0.087*(SPACECRAFT_MASS_BEFORE_APOGEE_BURN);--ADA
PTOR);
  NEW_LINE(2);
  SET_COL(10);
  PUT(" Structural mass is  ");
  SET_COL(60);
  PUT(STRUCTURAL_MASS, FORE => 6, AFT => 2, EXP => 0);
  PUT(" kgs");




  SPACECRAFT_BOL_MASS:=SPACECRAFT_MASS_BEFORE_APOGEE_BURN
          -AMF-ADAPTOR-PRE_AMF;
  NEW_LINE(2);
  SET_COL(10);
  PUT(" Spacecraft beginning of life mass is");
  SET_COL(60);
  PUT(SPACECRAFT_BOL_MASS, FORE => 6, AFT => 2, EXP => 0);
  PUT(" kgs");

  if DRUM_SPINNER = FALSE then
    THERMAL_CONTROL_MASS:=0.032*SPACECRAFT_BOL_MASS;
    NEW_LINE(2);
    SET_COL(10);
    PUT(" Thermal control mass is  ");
    SET_COL(60);
    PUT(THERMAL_CONTROL_MASS, FORE => 6, AFT => 2, EXP => 0);
    PUT(" kgs");
  else
    THERMAL_CONTROL_MASS:=0.027*SPACECRAFT_BOL_MASS;
    NEW_LINE(2);
    SET_COL(10);
    PUT(" Thermal control mass is  ");
    SET_COL(60);
    PUT(THERMAL_CONTROL_MASS, FORE => 6, AFT => 2, EXP => 0);
    PUT(" kgs");
end if;

if DRUM_SPINNER = FALSE then
    ATTITUDE_CONTROL:=65.0+0.022*(SPACECRAFT_BOL_MASS-700.0);
    NEW_LINE(2);
    SET_COL(10);
    PUT(" Attitude_control_mass is ");
    SET_COL(60);
    PUT(ATTITUDE_CONTROL, FORE => 6, AFT => 2, EXP => 0);
    PUT(" kgs");
else
    ATTITUDE_CONTROL:=31.0+0.027*(SPACECRAFT_BOL_MASS-700.0);
    NEW_LINE(2);
    SET_COL(10);
    PUT(" Attitude_control_mass is ");
    SET_COL(60);
    PUT(ATTITUDE_CONTROL, FORE => 6, AFT => 2, EXP => 0);
    PUT(" kgs");
end if;

ELECTRICAL_SYSTEM_MASS:=0.039*SPACECRAFT_BOL_MASS;
NEW_LINE(2);
SET_COL(10);
PUT(" Electrical system mass is ");
SET_COL(60);
PUT(ELECTRICAL_SYSTEM_MASS, FORE => 6, AFT => 2, EXP => 0);
PUT(" kgs");



MECHANICAL_SYSTEM_MASS:=0.014*SPACECRAFT_BOL_MASS;
NEW_LINE(2);
SET_COL(10);
PUT(" Mechanical system mass is ");
SET_COL(60);
PUT(MECHANICAL_SYSTEM_MASS, FORE => 6, AFT => 2, EXP => 0);
PUT(" kgs");

BYPROPELLENT_MASS:=0.084*PROPELLENT_EXPENDITURE;
NEW_LINE(2);
SET_COL(10);
PUT(" Propellent is ");
SET_COL(60);
PUT(BYPROPELLENT_MASS , FORE => 6, AFT => 2, EXP => 0);
PUT(" kgs");
NEW_LINE(2);
```

```
  PROPELLENT_PRESSURANT_MASS:=PROPELLENT_EXPENDITURE-AMF;
  SET_COL(10);
  PUT("Mass of propellent pressurant is ");
  SET_COL(60);
  PUT(PROPELLENT_PRESSURANT_MASS,FORE => 6, AFT => 2, EXP => 0);
  PUT(" kgs");
  NEW_LINE(2);
  SPACECRAFT_DRY_MASS:=SPACECRAFT_MASS_BEFORE_APOGEE_BURN
          -PROPELLENT_EXPENDITURE;
  SET_COL(10);
  PUT("Spacecraft Dry Mass is");
  SET_COL(60);
  PUT(SPACECRAFT_DRY_MASS,FORE => 6, AFT => 2, EXP => 0);
  PUT(" kgs");
  NEW_LINE(2);

  SET_COL(10); PUT("TO CONTINUE ENTER ANY INTEGER");
GET_INTEGER(I);
  VIDEO.CLEAR_SCREEN;

  MASS_MARGIN:=0.1*SPACECRAFT_DRY_MASS;
  new_line(1);
  SET_COL(10);
  PUT("Spacecraft mass margin is");
  SET_COL(50);
  PUT(MASS_MARGIN,FORE => 6, AFT => 2, EXP => 0);
  PUT(" kgs");
  NEW_LINE(2);

  COMM_PACKAGE_MASS:= SPACECRAFT_MASS_BEFORE_APOGEE_BURN
            -STRUCTURAL_MASS
            -THERMAL_CONTROL_MASS
            -PROPELLENT_EXPENDITURE
            -ATTITUDE_CONTROL
            -ELECTRICAL_SYSTEM_MASS
            -MECHANICAL_SYSTEM_MASS
            -MASS_MARGIN
            -MASS_CHANGE_POST_AMF
            -PROPELLENT_PRESSURANT_MASS;

  SET_COL(10);
  PUT("Communications package mass is ");
  SET_COL(50);
  PUT(COMM_PACKAGE_MASS,FORE => 6, AFT => 2, EXP => 0);
  PUT(" kgs");
  NEW_LINE(2);

PUT_LINE("*****************************************************
***********");
  NEW_LINE(2);
```

```
--*********************************************************

   CREATE(OUTF,NAME=>"PROPBUDG.DAT");

  -- header for propellent budget
  -- top line of header
  SET_LINE(OUTF,1);
  SET_COL(OUTF,1);
  PUT(OUTF,"MANEUVER");
  SET_COL(OUTF,25);
  PUT(OUTF,"Delta");
  SET_COL(OUTF,35);
  PUT(OUTF,"Specific");
  SET_COL(OUTF,48);
  PUT(OUTF,"Mass");
  SET_COL(OUTF,58);
  PUT(OUTF,"Final");
  SET_COL(OUTF,66);
  PUT(OUTF,"Efficiency");

  -- second line of header
  SET_LINE(OUTF,2);
  SET_COL(OUTF,23);
  PUT(OUTF,"Velocity");
  SET_COL(OUTF,35);
  PUT(OUTF,"Impulse");
  SET_COL(OUTF,47);
  PUT(OUTF,"Change");
  SET_COL(OUTF,58);
  PUT(OUTF,"Mass");

  -- third line of header
  SET_LINE(OUTF,3);
  SET_COL(OUTF,25);
  PUT(OUTF,"(m/s)");
  SET_COL(OUTF,37);
  PUT(OUTF,"(s)");
  SET_COL(OUTF,48);
  PUT(OUTF,"(kg)");
  SET_COL(OUTF,58);
  PUT(OUTF,"(kg)");

  SET_LINE(OUTF,4);

PUT("-----------------------------------------------------------
----");

  -- start first data line
  SET_LINE(OUTF,5);
  SET_COL(OUTF,1);
  PUT(OUTF,"Separation");
  SET_COL(OUTF,57);

PUT(OUTF,SPACECRAFT_MASS_BEFORE_APOGEE_BURN,FORE=>4,AFT=>1,EXP=>0
);
  -- second line of data
  SET_LINE(OUTF,7);
  SET_COL(OUTF,1);
  PUT(OUTF,"Before AMF");
  SET_COL(OUTF,46);
  PUT(OUTF,ADAPTOR,FORE=>4,AFT=>1,EXP=>0);
  SET_COL(OUTF,57);

PUT(OUTF,SPACECRAFT_MASS_BEFORE_APOGEE_BURN-ADAPTOR,FORE=>4,AFT=>
1,EXP=>0);
  -- third line of data
  SET_LINE(OUTF,9);
  SET_COL(OUTF,1);
  PUT(OUTF,"AMF");
  SET_COL(OUTF,25);
  PUT(OUTF,DELTA_VELOCITY*1000.0,FORE=>4,AFT=>1,EXP=>0);
  SET_COL(OUTF,37);
  PUT(OUTF,APOGEE_MOTOR_IMPULSE,FORE=>4,AFT=>1,EXP=>0);
  SET_COL(OUTF,46);
  PUT(OUTF,AMF,FORE=>4,AFT=>1,EXP=>0);
  SET_COL(OUTF,57);
  PUT(OUTF,SPACECRAFT_MASS_BEFORE_APOGEE_BURN-ADAPTOR-AMF
                    ,FORE=>4,AFT=>1,EXP=>0);
  SET_COL(OUTF,69);
  PUT(OUTF,"1.00");

  -- fourth line of data
  SET_LINE(OUTF,11);
  SET_COL(OUTF,1);
  PUT(OUTF,"Post AMF");
  SET_COL(OUTF,46);
  PUT(OUTF,MASS_CHANGE_POST_AMF,FORE=>4,AFT=>1,EXP=>0);
  SET_COL(OUTF,57);
  PUT(OUTF,SPACECRAFT_MASS_BEFORE_APOGEE_BURN-ADAPTOR
       -AMF-PRE_AMF-MASS_CHANGE_POST_AMF,FORE=>4,AFT=>1,EXP=>0);
  -- fifth line of data ns station keeping
  SET_LINE(OUTF,13);
  SET_COL(OUTF,1);
  PUT(OUTF,"N-S station keeping");
  SET_COL(OUTF,25);
  PUT(OUTF,DELTA_VELOCITY_NORTH_SOUTH,FORE=>4,AFT=>1,EXP=>0);
  SET_COL(OUTF,37);
  PUT(OUTF,ORBIT_IMPULSE,FORE=>4,AFT=>1,EXP=>0);
  SET_COL(OUTF,46);
  PUT(OUTF,MASS_NS_STATION_KEEPING,FORE=>4,AFT=>1,EXP=>0);
  SET_COL(OUTF,57);
```

```
PUT(OUTF,SPACECRAFT_MASS_BEFORE_APOGEE_BURN-ADAPTOR
     -AMF-PRE_AMF-MASS_CHANGE_POST_AMF
     -MASS_NS_STATION_KEEPING,FORE=>4,AFT=>1,EXP=>0);
SET_COL(OUTF,69);
PUT(OUTF,EFF_NS,FORE=>1,AFT=>2,EXP=>0);

-- sixth line of data ew station keeping
SET_LINE(OUTF,15);
SET_COL(OUTF,1);
PUT(OUTF,"E-W station keeping");
SET_COL(OUTF,25);
PUT(OUTF,DELTA_VELOCITY_EAST_WEST,FORE=>4,AFT=>1,EXP=>0);
SET_COL(OUTF,37);
PUT(OUTF,ORBIT_IMPULSE,FORE=>4,AFT=>1,EXP=>0);
SET_COL(OUTF,46);
PUT(OUTF,MASS_EW_STATION_KEEPING,FORE=>4,AFT=>1,EXP=>0);
SET_COL(OUTF,57);
PUT(OUTF,SPACECRAFT_MASS_BEFORE_APOGEE_BURN-ADAPTOR
     -AMF-PRE_AMF-MASS_CHANGE_POST_AMF
     -MASS_NS_STATION_KEEPING-MASS_EW_STATION_KEEPING
     ,FORE=>4,AFT=>1,EXP=>0);
SET_COL(OUTF,69);
PUT(OUTF,EFF_EW,FORE=>1,AFT=>2,EXP=>0);

-- seventh line of data  station repositioning
SET_LINE(OUTF,17);
SET_COL(OUTF,1);
PUT(OUTF,"Station Repositioning");
SET_COL(OUTF,25);

PUT(OUTF,DELTA_VELOCITY_STATION_REPOSITIONING,FORE=>4,AFT=>1,EXP=
>0);
SET_COL(OUTF,37);
PUT(OUTF,ORBIT_IMPULSE,FORE=>4,AFT=>1,EXP=>0);
SET_COL(OUTF,46);
PUT(OUTF,MASS_STATION_REPOSITIONING,FORE=>4,AFT=>1,EXP=>0);
SET_COL(OUTF,57);
PUT(OUTF,SPACECRAFT_MASS_BEFORE_APOGEE_BURN-ADAPTOR
     -AMF-PRE_AMF-MASS_CHANGE_POST_AMF
     -MASS_NS_STATION_KEEPING-MASS_EW_STATION_KEEPING
     -MASS_STATION_REPOSITIONING,FORE=>4,AFT=>1,EXP=>0);
SET_COL(OUTF,69);
PUT(OUTF,EFF_STATION_REPOSITIONING,FORE=>1,AFT=>2,EXP=>0);

-- eight line of data attitude control
SET_LINE(OUTF,19);
SET_COL(OUTF,1);
PUT(OUTF,"Attitude Control");
SET_COL(OUTF,46);
PUT(OUTF,ON_ORBIT_CONTROL,FORE=>4,AFT=>1,EXP=>0);
SET_COL(OUTF,57);
PUT(OUTF,SPACECRAFT_MASS_BEFORE_APOGEE_BURN-ADAPTOR
      -AMF-PRE_AMF-MASS_CHANGE_POST_AMF
      -MASS_NS_STATION_KEEPING-MASS_EW_STATION_KEEPING
      -MASS_STATION_REPOSITIONING-ON_ORBIT_CONTROL
      ,FORE=>4,AFT=>1,EXP=>0);
SET_COL(OUTF,69);
PUT(OUTF,EFF_NS,FORE=>1,AFT=>2,EXP=>0);

 -- ninth line of data de-orbit
SET_LINE(OUTF,21);
SET_COL(OUTF,1);
PUT(OUTF,"De-orbit");
SET_COL(OUTF,25);
PUT(OUTF,DELTA_VELOCITY_DEORBIT,FORE=>4,AFT=>1,EXP=>0);
SET_COL(OUTF,37);
PUT(OUTF,ORBIT_IMPULSE,FORE=>4,AFT=>1,EXP=>0);
SET_COL(OUTF,46);
PUT(OUTF,MASS_DEORBIT,FORE=>4,AFT=>1,EXP=>0);
SET_COL(OUTF,57);
PUT(OUTF,SPACECRAFT_MASS_BEFORE_APOGEE_BURN-ADAPTOR
      -AMF-PRE_AMF-MASS_CHANGE_POST_AMF
      -MASS_NS_STATION_KEEPING-MASS_EW_STATION_KEEPING
      -MASS_STATION_REPOSITIONING-ON_ORBIT_CONTROL-MASS_DEORBIT
      ,FORE=>4,AFT=>1,EXP=>0);
SET_COL(OUTF,69);
PUT(OUTF,EFF_DEORBIT,FORE=>1,AFT=>2,EXP=>0);

  -- tenth line of data pressurant
SET_LINE(OUTF,23);
SET_COL(OUTF,1);
PUT(OUTF,"Pressurant");
SET_COL(OUTF,46);
PUT(OUTF,PRESSURANT,FORE=>4,AFT=>1,EXP=>0);
SET_COL(OUTF,57);
PUT(OUTF,SPACECRAFT_MASS_BEFORE_APOGEE_BURN-ADAPTOR
      -AMF-PRE_AMF-MASS_CHANGE_POST_AMF
      -MASS_NS_STATION_KEEPING-MASS_EW_STATION_KEEPING
      -MASS_STATION_REPOSITIONING-MASS_DEORBIT-PRESSURANT
      ,FORE=>4,AFT=>1,EXP=>0);

  -- eleventh line of data  margin propellent
SET_LINE(OUTF,25);
SET_COL(OUTF,1);
PUT(OUTF,"Margin Propellent");
SET_COL(OUTF,46);
PUT(OUTF,PROPELLENT_MARGIN,FORE=>4,AFT=>1,EXP=>0);
SET_COL(OUTF,57);
PUT(OUTF,SPACECRAFT_MASS_BEFORE_APOGEE_BURN-ADAPTOR
      -AMF-PRE_AMF-MASS_CHANGE_POST_AMF
      -MASS_NS_STATION_KEEPING-MASS_EW_STATION_KEEPING
      -MASS_STATION_REPOSITIONING-MASS_DEORBIT-PRESSURANT
      -PROPELLENT_MARGIN,FORE=>4,AFT=>1,EXP=>0);
-- total mass change
```

```
 SET_LINE(OUTF,26);
 SET_COL(OUTF,1);

PUT(OUTF,"______________________________________________________________
_____________________");
 SET_LINE(OUTF,27);
 SET_COL(OUTF,1);
 PUT(OUTF,"Total Mass Change");
 SET_COL(OUTF,46);
 PUT(OUTF,ADAPTOR+AMF+PRE_AMF+MASS_CHANGE_POST_AMF
      +MASS_NS_STATION_KEEPING+MASS_EW_STATION_KEEPING
      +MASS_STATION_REPOSITIONING+MASS_DEORBIT+PRESSURANT
      +PROPELLENT_MARGIN,FORE=>4,AFT=>1,EXP=>0);


 CLOSE(OUTF);

--*********************************************************
 NEW_LINE(1);
 SET_LINE(OUTM,1);
 SET_COL(OUTM,10);
 PUT(OUTM,"SPACECRAFT  MASS  SUMMARY");
 SET_LINE(OUTM,2);

PUT(OUTM,"------------------------------------------------------
---------");
 SET_LINE(OUTM,3);
 PUT(OUTM,"Subsystem");
 SET_COL(OUTM,40);
 PUT(OUTM,"Mass (kg)");
 SET_LINE(OUTM,4);

PUT(OUTM,"------------------------------------------------------
---------");


 -- Structure
 SET_LINE(OUTM,6);
 PUT(OUTM,"Structure");
 SET_COL(OUTM,40);
 PUT(OUTM,STRUCTURAL_MASS,FORE=>4,AFT=>1,EXP=>0);

 -- Thermal
 SET_LINE(OUTM,8);
 PUT(OUTM,"Thermal");
 SET_COL(OUTM,40);
 PUT(OUTM,THERMAL_CONTROL_MASS,FORE=>4,AFT=>1,EXP=>0);

 -- Propulsion
 SET_LINE(OUTM,10);
 PUT(OUTM,"Propulsion");
 SET_COL(OUTM,40);
```

```
PUT(OUTM,BYPROPELLENT_MASS,FORE=>4,AFT=>1,EXP=>0);



-- Attitude Control
SET_LINE(OUTM,12);
PUT(OUTM,"Attitude Control");
SET_COL(OUTM,40);
PUT(OUTM,ATTITUDE_CONTROL,FORE=>4,AFT=>1,EXP=>0);

-- Electric Integration
SET_LINE(OUTM,14);
PUT(OUTM,"Electric Integration");
SET_COL(OUTM,40);
PUT(OUTM,ELECTRICAL_SYSTEM_MASS,FORE=>4,AFT=>1,EXP=>0);

-- Mechanical Integration
SET_LINE(OUTM,16);
PUT(OUTM,"Mechanical Integration");
SET_COL(OUTM,40);
PUT(OUTM,MECHANICAL_SYSTEM_MASS,FORE=>4,AFT=>1,EXP=>0);

-- Mass Margin
SET_LINE(OUTM,18);
PUT(OUTM,"Mass Margin");
SET_COL(OUTM,40);
PUT(OUTM,MASS_MARGIN,FORE=>4,AFT=>1,EXP=>0);

-- Dry Spacecraft Mass
SET_LINE(OUTM,20);
PUT(OUTM,"Dry Spacecraft Mass");
SET_COL(OUTM,40);
PUT(OUTM,SPACECRAFT_DRY_MASS,FORE=>4,AFT=>1,EXP=>0);

-- Propellent Pressurant
SET_LINE(OUTM,22);
PUT(OUTM,"Propellent Pressurant");
SET_COL(OUTM,40);
PUT(OUTM,PROPELLENT_PRESSURANT_MASS,FORE=>4,AFT=>1,EXP=>0);

-- Apogee Motor Expendable
SET_LINE(OUTM,24);
PUT(OUTM,"Apogee Motor Expendable");
SET_COL(OUTM,40);
PUT(OUTM,AMF,FORE=>4,AFT=>1,EXP=>0);

-- Spacecraft Mass at Separation
SET_LINE(OUTM,26);
PUT(OUTM,"Spacecraft Mass at Seperation");
SET_COL(OUTM,40);

PUT(OUTM,SPACECRAFT_MASS_BEFORE_APOGEE_BURN,FORE=>4,AFT=>1,EXP=>0
);

--*************************************************************
**********


end MASS;

procedure ELECTRICAL_SYSTEM
        (SPACECRAFT_MASS_BEFORE_APOGEE_BURN    : in
FLOAT;
         DRUM_SPINNER                       : in out
BOOLEAN;
         COMM_PACKAGE_MASS                       : in
FLOAT) is


  NO,
  N,
  Y,
  YES,
  CHAR              : CHARACTER;


  SOLAR_ARRAY,                                 -- kg
  CHARGE_ARRAY,                                 -- kg
  SHUNT,                                   -- kg
  CHARGE_CONTROL,                                 -- kg
  BATTERY,                                  -- kg
  DISCHARGE_REGULATOR : FLOAT;                          -- kg
  LIFE_FACTOR: FLOAT                := 1.05;      --
  POWER_MARGIN : FLOAT                := 1.1;       --
margin for error
  TTC_FACTOR : FLOAT                 := 1.75;      -- TT&C
scale factor
  TRACKING_TELEMETRY_REFERENCE : FLOAT    := 28.0;      --
intelsat v
  INTELSAT_7_REFERENCE       : FLOAT   :=3445.0;
  INTELSAT_6_REFERENCE       : FLOAT   :=3700.0;
  INTELSAT_5_REFERENCE       : FLOAT   :=1900.0;
  INTELSAT_7_ANTENNA_MASS     : FLOAT   :=70.0;
  INTELSAT_6_ANTENNA_MASS     : FLOAT   :=309.0;
  INTELSAT_5_ANTENNA_MASS     : FLOAT   :=59.0;
  INTELSAT_5_HOUSEKEEPING_POWER: constant FLOAT := 211.0; --
intelsat V
  INTELSAT_6_HOUSEKEEPING_POWER: constant FLOAT := 347.0; --
intelsat VI
  INTELSAT_7_HOUSEKEEPING_POWER: constant FLOAT := 613.0; --
intelsat VII

  ELECTRICAL_POWER_MASS,
  COMM_ELECTRICAL_SUBSYSTEM_MASS,
  BATTERY_LOAD,
  PAYLOAD_POWER,
  PAYLOAD_MASS,
  ANTENNA_MASS,
  REFERENCE,
  HOUSEKEEPING_POWER,
  SOLAR_ARRAY_LOAD,
  POWER_FACTOR,
  X,

  TRACKING_TELEMETRY : FLOAT;

  CHOICE             : INTEGER;


-- Reads an integer input from the keyboard


begin
  REFERENCE:=INTELSAT_5_REFERENCE;
  ANTENNA_MASS:=INTELSAT_5_ANTENNA_MASS;
  HOUSEKEEPING_POWER:=INTELSAT_5_HOUSEKEEPING_POWER;

-- The mass of the electrical power system is
  NEW_LINE(2);
  SET_COL(10);
  PUT_LINE("Enter the POWER requirements of the");
  SET_COL(10);
  PUT_LINE("Spacecraft in watts as a real number");
  SET_COL(15);
  GET_DATA(PAYLOAD_POWER);
  VIDEO.CLEAR_SCREEN;
  NEW_LINE(2);
  SET_COL(15);
  PUT("Payload power requirements are ");
  SET_COL(60);

  PUT(PAYLOAD_POWER, FORE => 6, AFT => 2, EXP => 0);
  NEW_LINE(2);

PUT_LINE("********************************************************
***********");
  NEW_LINE(2);
  SET_COL(5);
  PUT_LINE("Choose which satellite you want as your reference
for ");
  SET_COL(5);
  PUT_LINE("housekeeping power, mass in kilograms, and Antenna
Mass ");
  SET_COL(5);
  PUT_LINE("          Intelsat V    Intelsat VI    Intelsat
VII");
```

```
  SET_COL(5);
  PUT_LINE("Mass           1900.0 kgs      3700.0 kgs
3445.0 kgs");
  SET_COL(5);
  PUT_LINE("Antenna Mass      59.0 kgs       309.0 kgs
75.0 kgs");
  SET_COL(5);
  PUT_LINE("Housekeeping    211.0 Watts    347.2 Watts
613.0 Watts");
  PUT_LINE("Power");
  SET_COL(5);

PUT_LINE("*****************************************************
***********");
  SET_COL(5);
  PUT_LINE("For an INTELSAT V reference enter an integer  '1'
");
  SET_COL(5);
  PUT_LINE("For an INTELSAT VI  reference enter an integer  '2'
");
  SET_COL(5);
  PUT_LINE("For an INTELSAT VII  reference enter an integer  '3'
");
  SET_COL(5);
  PUT_LINE("TO USE YOUR OWN  REFERENCE VALUES enter an integer
'4'  ");
  GET_INTEGER(CHOICE);

  case CHOICE is
    when 1=>
    REFERENCE:=INTELSAT_5_REFERENCE;
    ANTENNA_MASS:=INTELSAT_5_ANTENNA_MASS;
    HOUSEKEEPING_POWER:=INTELSAT_5_HOUSEKEEPING_POWER;


    when 2=>
    REFERENCE:=INTELSAT_6_REFERENCE;
    ANTENNA_MASS:=INTELSAT_6_ANTENNA_MASS;
    HOUSEKEEPING_POWER:=INTELSAT_6_HOUSEKEEPING_POWER;

    when 3=>
    REFERENCE:=INTELSAT_7_REFERENCE;
    ANTENNA_MASS:=INTELSAT_7_ANTENNA_MASS;
    HOUSEKEEPING_POWER:=INTELSAT_7_HOUSEKEEPING_POWER;
    when 4=>
    PUT("Enter Satellite Mass Reference");
    SET_COL(15);
    GET_DATA(REFERENCE);
    NEW_LINE(3);

    PUT("Enter Satellite Antenna Mass Reference");
    SET_COL(15);
    GET_DATA(ANTENNA_MASS);
    NEW_LINE(3);

    PUT("Enter Satellite Housekeeping Power Requirements
Reference");
    SET_COL(15);
    GET_DATA(HOUSEKEEPING_POWER);
    NEW_LINE(3);


    when OTHERS =>
    NEW_LINE(2);
    SET_COL(5);
    PUT("Understand INTELSAT V DATA WILL BE USED");
  end case;



  VIDEO.CLEAR_SCREEN;

PUT_LINE("*****************************************************
***********");
  NEW_LINE;
  HOUSEKEEPING_POWER:=(SPACECRAFT_M  SS_BEFORE_APOGEE_BURN
            /REFERENCE)
            *HOUSEKEEPING_POWER;
  NEW_LINE(2);
  SET_COL(10);
  PUT(" Housekeeping power is  ");
  SET_COL(50);
  PUT(HOUSEKEEPING_POWER, FORE => 6, AFT => 2, EXP => 0);
  PUT("  kg  ),

  BATTERY_  AD:=(HOUSEKEEPING_POWER+PAYLOAD_POWER)*LIFE_FACTOR;
  NEW_LINE( );
  SE  COL(10);
  PUT(" Battery load is  ");
  SET_COL(50);
  PUT(BATTERY_LOAD, FORE => 6, AFT => 2, EXP => 0);
  PUT("  kg");




  SOLAR_ARRAY_LOAD:=BATTERY_LOAD*POWER_MARGIN;
  NEW_LINE(2);
  SET_COL(10);
  PUT(" Solar array load is  ");
  SET_COL(50);
  PUT(SOLAR_ARRAY_LOAD, FORE => 6, AFT => 2, EXP => 0);
  PUT("  kg");
```

```
--ELECTRICAL_POWER_MASS:=(PAYLOAD_POWER+HOUSEKEEPING_POWER)*(BATT
TERY_LOAD+SOLAR_ARRAY_LOAD);


  if DRUM_SPINNER = FALSE then

    if BATTERY_LOAD < 1875.0  then
    SOLAR_ARRAY                      := 50.0;        -- kg
    CHARGE_ARRAY                     := 7.8;         -- kg
    SHUNT                        := 7.5;         -- kg
    CHARGE_CONTROL                   := 1.5;         -- kg
    BATTERY                      := 56.8;        -- kg
    DISCHARGE_REGULATOR              := 0.2;         -- kg

    elsif BATTERY_LOAD < 3125.0 then
    SOLAR_ARRAY                      := 42.0;        -- kg
    CHARGE_ARRAY                     := 6.6;         -- kg
    SHUNT                        := 7.5;         -- kg
    CHARGE_CONTROL                   := 1.5;         -- kg
    BATTERY                      := 47.3;        -- kg
    DISCHARGE_REGULATOR              := 0.2;         -- kg

    elsif BATTERY_LOAD <  4375.0 then
    SOLAR_ARRAY                      := 33.0;        -- kg
    CHARGE_ARRAY                     := 5.1;         -- kg
    SHUNT                        := 7.5;         -- kg
    CHARGE_CONTROL                   := 1.5;         -- kg
    BATTERY                      := 47.3;        -- kg
    DISCHARGE_REGULATOR              := 0.2;         -- kg

    elsif BATTERY_LOAD >=  4375.0 then
    SOLAR_ARRAY                      := 25.0;        -- kg
    CHARGE_ARRAY                     := 3.9;         -- kg
    SHUNT                        := 7.5;         -- kg
    CHARGE_CONTROL                   := 1.5;         -- kg
    BATTERY                      := 47.3;        -- kg
    DISCHARGE_REGULATOR              := 0.2;         -- kg
   end if;

  elsif BATTERY_LOAD < 1875.0 then
    SOLAR_ARRAY                      := 125.0:       -- kg
    CHARGE_ARRAY                     := 19.5;        -- kg
    SHUNT                        := 7.5;         -- kg
    CHARGE_CONTROL                   := 1.5;         -- kg
    BATTERY                      := 56.8;        -- kg
    DISCHARGE_REGULATOR              := 0.2;         -- kg

  else
    SOLAR_ARRAY                      := 70.0;        -- kg
    CHARGE_ARRAY                     := 11.0;        -- kg
    SHUNT                         := 7.5;       -- kg
    CHARGE_CONTROL                     := 1.5;       -- kg
    BATTERY                       := 47.3;      -- kg
    DISCHARGE_REGULATOR                := 0.2;       -- kg
  end if;


  POWER_FACTOR:=  LIFE_FACTOR

*(POWER_MARGIN*SOLAR_ARRAY+POWER_MARGIN*CHARGE_ARRAY
          +SHUNT+CHARGE_CONTROL+BATTERY+DISCHARGE_REGULATOR)
          *0.001;
  NEW_LINE(2);
  SET_COL(10);
  PUT(" Power Factor is  ");
  SET_COL(50);
  PUT(POWER_FACTOR, FORE => 4, AFT => 4, EXP => 0);



ELECTRICAL_POWER_MASS:=(PAYLOAD_POWER+HOUSEKEEPING_POWER)*POWER_F
ACTOR;
  NEW_LINE(2);
  SET_COL(10);
  PUT(" Electrical power sub-system mass is  ");
  SET_COL(50);
  PUT(ELECTRICAL_POWER_MASS, FORE => 6, AFT => 2, EXP => 0);
  PUT("  kg");

  PAYLOAD_MASS:= COMM_PACKAGE_MASS-ELECTRICAL_POWER_MASS;
  NEW_LINE(2);
  SET_COL(10);
  PUT(" Payload mass is  ");
  SET_COL(50);
  PUT(PAYLOAD_MASS, FORE => 6, AFT => 2, EXP => 0);
  PUT("  kg");


  TRACKING_TELEMETRY:=
(SPACECRAFT_MASS_BEFORE_APOGEE_BURN/REFERENCE)
          *TRACKING_TELEMETRY_REFERENCE;
  NEW_LINE(2);
  SET_COL(10);
  PUT(" Tracking telemetry and control mass is  ");
  SET_COL(50);
  PUT(TRACKING_TELEMETRY, FORE => 6, AFT => 2, EXP => 0);
  PUT("  kg");
--****************************************************************
*****
  -- communications
  SET_LINE(OUTM,28);
```

```
 PUT(OUTM,"Communications");                   --*
 SET_COL(OUTM,40);
 PUT(OUTM,PAYLOAD_MASS,FORE=>4,AFT=>1,EXP=>0);

 -- Antennas
 SET_LINE(OUTM,30);                          --*
 PUT(OUTM,"Antenna Reference Mass");
 SET_COL(OUTM,40);
 PUT(OUTM,ANTENNA_MASS,FORE=>4,AFT=>1,EXP=>0);

 -- electric power subsystem mass
 SET_LINE(OUTM,32);
 PUT(OUTM,"Electric Power");                  --*
 SET_COL(OUTM,40);
 PUT(OUTM,ELECTRICAL_POWER_MASS,FORE=>4,AFT=>1,EXP=>0);

 -- Telemetry and Control
 SET_LINE(OUTM,34);
 PUT(OUTM,"Telemetry and Command");              --*
 SET_COL(OUTM,40);
 PUT(OUTM,TRACKING_TELEMETRY,FORE=>4,AFT=>1,EXP=>0);


--****************************************************************
*******



end ELECTRICAL_SYSTEM;




----------------------------------------------------------------
----
----------------------------------------------------------------
----
begin
  CREATE(OUTM,NAME=>"SYSMASS.DAT");

  DUAL_SPIN                    (DRUM_SPINNER);

  VELOCITY                     (INCLINATION_RADIANS,
                DELTA_VELOCITY);

  STATION_KEEPING_REPOSITIONING   (DELTA_VELOCITY_NORTH_SOUTH,
                DELTA_VELOCITY_EAST_WEST,
                DELTA_VELOCITY_STATION_REPOSITIONING,
                EFF_NS,
                EFF_EW);

  MASS                       (DELTA_VELOCITY_NORTH_SOUTH,
                  DELTA_VELOCITY_EAST_WEST,
                  DELTA_VELOCITY_STATION_REPOSITIONING,
                  DELTA_VELOCITY,
                  EFF_NS,
                  EFF_EW,
                  COMM_PACKAGE_MASS,
                  SPACECRAFT_MASS_BEFORE_APOGEE_BURN);

  ELECTRICAL_SYSTEM            (COMM_PACKAGE_MASS,
                  DRUM_SPINNER,
                  SPACECRAFT_MASS_BEFORE_APOGEE_BURN);


  CLOSE(OUTM);

  SET_COL(10);PUT("TO CONTINUE ENTER ANY INTEGER");
GET_INTEGER(I);
  VIDEO.CLEAR_SCREEN;
  NEW_LINE(2);
  PUT_LINE("DATA FOR THIS DESIGN RUN ARE LOCATED IN THE
FOLLOWING FILES:");
  NEW_LINE(1);
  PUT_LINE("              SYSMASS.DAT");
  PUT_LINE("              PROPBUDG.DAT ");
  NEW_LINE(2);
  PUT_LINE("TO KEEP DATA FROM BEING ERASED ON NEXT RUN");
  PUT_LINE("USE DOS COMMAND REN (RENAME) ");
  NEW_LINE(1);
  PUT_LINE("EXAMPLE - REN SYSMASS.DAT   SYSMASS.XYZ");


end MASSPRO;
```

I. Electric Photovoltaic Power System

```
-- Title         : Solar Power Determination
-- Author         : David Lashbrook
-- Date          : 01 February 1992
-- Revised        : 12 May 1992
-- Compiler        : OPENADA EXT
-- Description   : This procedure determines the solar power and number of
--               cells required in to provide stated power requirements
--               for a geosynchronous orbit.

--with TEXT_IO, GENERIC_ELEMENTARY_FUNCTIONS, GETDATA,VIDEO;
--use  TEXT_IO, GETDATA ;

with TEXT_IO, MATH_LIB, GETDATA, VIDEO;
use  TEXT_IO, MATH_LIB, GETDATA;

procedure SOLARPOWER is
  package FLOAT_INOUT is new FLOAT_IO(FLOAT);
  use    FLOAT_INOUT;
  package INTEGER_INOUT is new INTEGER_IO(INTEGER);
  use    INTEGER_INOUT;
  package BOOLEAN_INOUT is new ENUMERATION_IO(BOOLEAN);
  use    BOOLEAN_INOUT;
--  package GEF_INOUT is new GENERIC_ELEMENTARY_FUNCTIONS(FLOAT);
--  use     GEF_INOUT;



BUS_VOLTAGE_ALLOWABLE_DEVIATION  : FLOAT:=0.5;



BYPASS_DIODE_VOLTAGE_DROP          : FLOAT:=1.1;   -- V_DD
-- V_D Open circuit failure of one cell + minimum battery discharge voltage at EOL

EOL_BATTERY_DISCHARGE_VOLTAGE       : FLOAT:=1.10;

DEPTH_OF_DISCHARGE                : FLOAT:=0.65;
ECLIPSE_TIME                   : FLOAT:=1.20;
MAXIMUM_BATTERY_CHARGE_VOLTAGE      : FLOAT:= 1.5;
SERIES_CONNECTED_DIODE_VOLTAGE_DROP : FLOAT:= 0.8;
NUMBER_SERIES_CONNECTED_DIODES      : FLOAT:= 3.0;   -- 3  assumed
BATTERY_CHARGER_VOLTAGE_DROP        : FLOAT:= 1.75;
CHARGE_DISCHARGE_EFFICIENCY_BATTERY : FLOAT:= 0.9;

EQUINOX_CHARGE_RATE             : constant FLOAT := 15.0; -- autumn
SOLSTICE_CHARGE_RATE            : constant FLOAT := 45.0; -- summer
PI                          : constant FLOAT := 3.14159265359;
SPACECRAFT_LIFE,
SPACECRAFT_MASS_BEFORE_APOGEE_BURN,
PAYLOAD_POWER,
CELL_AH,
BUS_POWER,
SERIES_CELLS_FOR_MIN_DIS_VOLT,  -- N
VOLTAGE_DROP_BATTERY_DC_DIODES, -- V_DD
VOLTAGE_BUS_LOW,                  -- LOWEST VOLTAGE ON BUS
CHARGING_VOLTAGE,                  -- MAXIMUM_CHARGE_VOLTAGE
BATTERY_CHARGING_VOLTAGE_DROP,  --
VOLTAGE_CHARGE_ARRAY,              -- voltage charge array
SERIES_CELLS_FOR_MIN_DIS_VOLT,
EFFICIENCY,
POWER_EQUINOX_CHARGE,
EQUINOX_CHARGE_TIME,
POWER_SOLSTICE_CHARGE,
SOLAR_ARRAY,
CHARGE_ARRAY,
SHUNT,
CHARGE_CONTROL,
DISCHARGE_REGULATOR,
PAYLOAD_POWER,
BATTERY_LOAD,
MINIMUM_DISCHARGE_BUS_VOLTAGE, -- V_DB
BUS_VOLTAGE,
NUMBER_OF_BUSES,
SOLAR_ARRAY_LOAD            : FLOAT;

LIFE_FACTOR                 : FLOAT   := 1.05;      --
POWER_MARGIN                   : FLOAT   := 1.1;      -- margin for error
INTELSAT_7_REFERENCE          : FLOAT   :=3445.0;    -- kgs
INTELSAT_6_REFERENCE          : FLOAT   :=2227.0;    -- kgs
INTELSAT_5_REFERENCE          : FLOAT   :=1900.0;    -- kgs
INTELSAT_7_HOUSEKEEPING_POWER: constant FLOAT := 613.0; -- intelsat VII
INTELSAT_6_HOUSEKEEPING_POWER: constant FLOAT := 347.0; -- intelsat VI
INTELSAT_5 HOUSEKEEPING_POWER: constant FLOAT := 211.0; -- intelsat V

X,
MASS_REFERENCE,
HOUSEKEEPING_POWER,
HOUSEKEEPING_POWER_REFERENCE    : FLOAT ;

FINAL                       : BOOLEAN :=TRUE;
DRUM_SPINNER                     : BOOLEAN := FALSE;

Y,
y,
N,
n,
CHAR                     : CHARACTER ;

I,
CHOICE,
N_INTEGER,
J                    : INTEGER ;

 procedure PRINT_HEADER is
   begin
     VIDEO.CLEAR_SCREEN;SET_LINE(1);
     NEW_LINE(2);
     SET_COL(10);
     PUT_LINE("This program walks through a basic design of the power");
     SET_COL(10);
     PUT_LINE("requirements of a solar powered geosynchronous satellite.");
     NEW_LINE;
     SET_COL(10);
     PUT_LINE("All pertinent data will be written to files.");
     SET_COL(10);
     PUT_LINE("CELPARAM.DAT     and     SOLCELL.DAT");
     new_line(1);

   end PRINT_HEADER;

procedure DUAL_SPIN (DRUM_SPINNER : in out BOOLEAN) is
begin
   SET_COL(10);
   PUT_LINE("Is your spacecraft Spin Stabilized ");
   SET_COL(15);
   GET_CHARACTER(char);
   if CHAR = 'Y' or CHAR = 'y' then
     DRUM_SPINNER:=TRUE;
     if DRUM_SPINNER = TRUE then
          VIDEO.CLEAR_SCREEN;SET_LINE(1);
          SET_COL(10);
          PUT_LINE("Satellite is Spin Stabilized");
          MINIMUM_DISCHARGE_BUS_VOLTAGE:=35.0;  -- V_DB
          BUS_VOLTAGE:=50.0;
          NEW_LINE(2);
          PUT_LINE("*********************************************************************");
     end if;
   else
     VIDEO.CLEAR_SCREEN;SET_LINE(1);
     SET_COL(10);
     PUT_LINE("Satellite is Three Axis Stabilized");
     MINIMUM_DISCHARGE_BUS_VOLTAGE:=30.0;  -- V_DB
     BUS_VOLTAGE:=42.0;
     NEW_LINE(2);
     PUT_LINE("*********************************************************************");
   end if;
end DUAL_SPIN;

procedure OPERATING_DATA  (BATTERY_LOAD                  : in out FLOAT;
                           DRUM_SPINNER                  : in out BOOLEAN;
                           MINIMUM_DISCHARGE_BUS_VOLTAGE : in out FLOAT;
                           BUS_VOLTAGE                   : in out FLOAT;
```

```
                              BYPASS_DIODE_VOLTAGE_DROP      : in out FLOAT;
                              EOL_BATTERY_DISCHARGE_VOLTAGE  : in out FLOAT;
                              BUS_POWER                      : in out FLOAT;
                              PAYLOAD_POWER                  : in out FLOAT;
                              DEPTH_OF_DISCHARGE             : in out FLOAT;
                              number_of_buses                : in out FLOAT;
                              SPACECRAFT_LIFE                : in out FLOAT;
                              ECLIPSE_TIME                   : in out FLOAT) is




  CHOICE,
  INPUT          : INTEGER ;

  REPLACE        : BOOLEAN := FALSE;

begin

 SET_COL(10);
 PUT_LINE("Enter the mass of the spacecraft in kilograms");
 NEW_LINE(2);
 SET_COL(10);
 GET_DATA(SPACECRAFT_MASS_BEFORE_APOGEE_BURN);
 VIDEO.CLEAR_SCREEN;
 SET_COL(15);
 PUT("Spacecraft mass before apogee motor burn is  ");
 PUT(SPACECRAFT_MASS_BEFORE_APOGEE_BURN, FORE => 6, AFT => 2, EXP => 0);
 PUT("  kgs");
 NEW_LINE(2);
 PUT_LINE("**********************************************************************");
 NEW_LINE(2);

  MASS_REFERENCE:=INTELSAT_5_REFERENCE;
  HOUSEKEEPING_POWER_REFERENCE:=INTELSAT_5_HOUSEKEEPING_POWER;

-- The mass of the electrical power system is
  SET_COL(10);
  PUT_LINE("Enter the POWER requirements of the Spacecraft in watts.");
  NEW_LINE(2);
  SET_COL(15);
  GET_DATA(PAYLOAD_POWER);
  VIDEO.CLEAR_SCREEN;
  NEW_LINE(2);
  PUT("Payload power requirements are ");
  SET_COL(50);
  PUT(PAYLOAD_POWER, FORE => 6, AFT => 2, EXP => 0);
  PUT("  Watts");
  NEW_LINE(2);
  PUT_LINE("**********************************************************************");
```

```
NEW_LINE(2);
SET_COL(5);
PUT_LINE("Choose which satellite you want as your reference for ");
SET_COL(5);
PUT_LINE("housekeeping power and spacecraft mass in kilograms.");
NEW_LINE(2);
PUT_LINE("******************************************************************");
NEW_LINE(1);
PUT_LINE("              '1'          '2'          '3'      ");
SET_COL(5);
PUT_LINE("            Intelsat V    Intelsat VI   Intelsat VII");
SET_COL(5);
PUT_LINE("Mass          1900.0 kgs    2227.0 kgs    3445.0 kgs");
SET_COL(5);
PUT_LINE("Housekeeping    211.0         347.0         613.0 ");
SET_COL(5);
PUT_LINE("Power" );
PUT_LINE("******************************************************************");
SET_COL(5);
PUT_LINE("For an INTELSAT V   reference    enter  integer  '1'  ");
SET_COL(5);
PUT_LINE("For an INTELSAT VI  reference    enter  integer  '2'  ");
SET_COL(5);
PUT_LINE("For an INTELSAT VII reference    enter  integer  '3'  ");
SET_COL(5);
PUT_LINE("For your own reference value's   enter  integer  '4'  ");

GET_INTEGER(CHOICE);

case CHOICE is
  when 1=>
      MASS_REFERENCE:=INTELSAT_5_REFERENCE;
      HOUSEKEEPING_POWER_REFERENCE:=INTELSAT_5_HOUSEKEEPING_POWER;


  when 2=>
      MASS_REFERENCE:=INTELSAT_6_REFERENCE;
      HOUSEKEEPING_POWER_REFERENCE:=INTELSAT_6_HOUSEKEEPING_POWER;

  when 3=>
      MASS_REFERENCE:=INTELSAT_7_REFERENCE;
      HOUSEKEEPING_POWER_REFERENCE:=INTELSAT_7_HOUSEKEEPING_POWER;

  when 4=>
      NEW_LINE(2);
      PUT_LINE("******************************************************************");
      NEW_LINE(2);
      VIDEO.CLEAR_SCREEN;
      PUT("Please enter desired REFERENCE  MASS");
      SET_COL(15);
      GET_DATA(MASS_REFERENCE);
      VIDEO.CLEAR_SCREEN;
```

```
        NEW_LINE(2);
        PUT_LINE("**********************************************************************");
        NEW_LINE(2);
        PUT("Please enter desired HOUSEKEEPING POWER reference");
        SET_COL(15);
        GET_DATA(HOUSEKEEPING_POWER_REFERENCE);
        VIDEO.CLEAR_SCREEN;
    when OTHERS =>
        NEW_LINE(2);
        SET_COL(5);
        PUT("Understand INTELSAT V DATA WILL BE USED");
end case;



HOUSEKEEPING_POWER:=(SPACECRAFT_MASS_BEFORE_APOGEE_BURN
                    /MASS_REFERENCE)
                    *HOUSEKEEPING_POWER_REFERENCE;
VIDEO.CLEAR_SCREEN;
NEW_LINE;
PUT("Housekeeping power is  ");
SET_COL(60);
PUT(HOUSEKEEPING_POWER, FORE => 6, AFT => 2, EXP => 0);

NEW_LINE(1);
BATTERY_LOAD:=(HOUSEKEEPING_POWER+PAYLOAD_POWER)*LIFE_FACTOR;
PUT("Battery load is (multplied by  life factor of 1.05)");
SET_COL(60);
PUT(BATTERY_LOAD, FORE => 6, AFT => 2, EXP => 0);

NEW_LINE(1);
SOLAR_ARRAY_LOAD:=BATTERY_LOAD*POWER_MARGIN;
PUT("Solar array load is (multiplied by power factor of 1.10) ");
SET_COL(60);
PUT(SOLAR_ARRAY_LOAD, FORE => 6, AFT => 2, EXP => 0);

new_line(2);
PUT_LINE("The Battery Load value will be used in future calculations.");
PUT_LINE("If you want to change this value enter a 'y' for YES. If you wish to ");
PUT_LINE("retain the value enter a 'n' for NO.  The value you enter  ");
put_line("should be the BATTERY_LOAD. );
PUT_LINE("BATTERY_LOAD = PAYLOAD POWER REQUIREMENTS + HOUSEKEEPING POWER");
PUT_LINE("This value is the BATTERY_LOAD and will have no design factors ");
PUT_LINE("applied to it.");
SET_COL(10);
NEW_LINE(2);
GET_CHARACTER(CHAR);
if CHAR = 'Y' or CHAR = 'y' then
   VIDEO.CLEAR_SCREEN;
   NEW_LINE(1);
   PUT("Payload power requirements are ");
```

```
    SET_COL(55);
    PUT(PAYLOAD_POWER, FORE => 6, AFT => 2, EXP => 0);
    PUT(" Watts");
    new_line(2);
    PUT("Calculated Housekeeping power is  ");
    SET_COL(55);
    PUT(HOUSEKEEPING_POWER, FORE => 6, AFT => 2, EXP => 0);
    PUT(" Watts");
    new_line(3);
    PUT("Please enter a value for the Battery Load");
    NEW_LINE(2);
    PUT("Remember to add your new housekeeping value to the payload power");
    NEW_LINE(3);
    GET_DATA(BATTERY_LOAD);
    NEW_LINE(2);
    PUT_LINE("**********************************************************************");
    NEW_LINE(2);
    VIDEO.CLEAR_SCREEN;
    PUT("Battery Load is  ");
    SET_COL(60);
    PUT(BATTERY_LOAD, FORE => 6, AFT => 2, EXP => 0);
    NEW_LINE(1);
    NEW_LINE(2);
    SOLAR_ARRAY_LOAD:=BATTERY_LOAD*POWER_MARGIN;
    PUT("Solar array load is (multiplied by power factor of 1.10) ");
    SET_COL(60);
    PUT(SOLAR_ARRAY_LOAD, FORE => 6, AFT => 2, EXP => 0);
  end if;

  NEW_LINE(2);
  PUT_LINE("**********************************************************************");
  NEW_LINE(2);

  PUT("Please enter the spacecraft life in years");
  NEW_LINE(2);
  SET_COL(10);
  GET_DATA(SPACECRAFT_LIFE);
  NEW_LINE(2);
  PUT("Spacecraft life is");
  PUT(SPACECRAFT_LIFE,FORE=>3,AFT=>1,EXP=>0);
  PUT(" years");
  NEW_LINE(2);
  STOP;


<<VALUE_NEW>>
  PUT_LINE("******************************************************************************");
  SET_COL(5);
  PUT_LINE("Default values for the following parameters are: ");
  PUT_LINE("******************************************************************************");
  SET_COL(5);
```

```
PUT("Minimum discharge voltage of bus          [1] ");
set_col(60);
PUT(MINIMUM_DISCHARGE_BUS_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);PUT(" volts");

NEW_LINE(1);
SET_COL(5);
PUT("DESIGN Satellite Bus Voltage          [2]");
SET_COL(60);
PUT(BUS_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);
PUT(" volts");

NEW_LINE(1);
SET_COL(5);
PUT("Bypass diode voltage drop          [3] ");
set_col(60);
PUT(BYPASS_DIODE_VOLTAGE_DROP,FORE=>4,AFT=>2,EXP=>0);PUT(" volts");

NEW_LINE(1);
SET_COL(5);
PUT("End of life battery discharge voltage     [4] ");
set_col(60);
PUT(EOL_BATTERY_DISCHARGE_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);PUT(" volts");

NEW_LINE(1);
SET_COL(5);
PUT("Satellite eclipse time hours          [5]  ");
SET_COL(60);
PUT(ECLIPSE_TIME,FORE=>4,AFT=>2,EXP=>0);
PUT(" hours");

NEW_LINE(1);
SET_COL(5);
PUT("Depth of Discharge          [6] ");
SET_COL(60);
PUT(DEPTH_OF_DISCHARGE,FORE=>4,AFT=>2,EXP=>0);

NEW_LINE(1);
SET_COL(5);
PUT("Maximum Battery Discharge Voltage       [7]");
SET_COL(60);
PUT(MAXIMUM_BATTERY_CHARGE_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);
PUT(" volts");

NEW_LINE(1);
SET_COL(5);
PUT("Series Connected Diode Voltage Drop      [8] ");
SET_COL(60);
PUT(SERIES_CONNECTED_DIODE_VOLTAGE_DROP,FORE=>4,AFT=>2,EXP=>0);
PUT(" volts");

NEW_LINE(1);
SET_COL(5);
```

```
PUT("Number of Series Connected Diodes          [9]  ");
SET_COL(60);
PUT(NUMBER_SERIES_CONNECTED_DIODES,FORE=>4,AFT=>2,EXP=>0);

NEW_LINE(1);
SET_COL(5);
PUT("Battery Charger Voltage Drop             [10] ");
SET_COL(60);
PUT(BATTERY_CHARGER_VOLTAGE_DROP,FORE=>4,AFT=>2,EXP=>0);
PUT("  volts");

NEW_LINE(1);
SET_COL(5);
PUT("Charge Discharge Voltage Drop            [11]");
SET_COL(60);
PUT(CHARGE_DISCHARGE_EFFICIENCY_BATTERY,FORE=>4,AFT=>2,EXP=>0);
new_LINE(2);

if REPLACE = FALSE then
  CHAR := N;
  PUT_LINE("If you desire to change any of the listed values please enter ");
  PUT_LINE("a 'y' for YES  otherwise enter a 'n' for NO");
  GET_CHARACTER(CHAR);
  if CHAR = 'Y' or CHAR = 'y' then
      CHAR := N;
      PUT_LINE("Enter number corresponding to value you wish to change.");
      set_col(10);
      GET_INTEGER(INPUT);
      VIDEO.CLEAR_SCREEN;
  else
      VIDEO.CLEAR_SCREEN;
      goto KEEP_VALUES;
  end if;
elsif REPLACE = TRUE then
  PUT_LINE("Enter number corresponding to value you wish to change.");
  set_col(10);
  GET_INTEGER(INPUT);
  VIDEO.CLEAR_SCREEN;
end if;

case INPUT is

  when 1=>
      VIDEO.CLEAR_SCREEN;
      NEW_LINE(2);
      PUT_LINE("*********************************************************************");
      NEW_LINE(2);
      PUT_LINE("Please enter the minimum satellite bus discharge voltage about ");
      PUT_LINE("30 volts for 3-axis stabilized and 35.0 for spin stabilized");
      NEW_LINE(2);
      SET_COL(10);
      GET_DATA(MINIMUM_DISCHARGE_BUS_VOLTAGE);
```

```
        PUT("Minimum discharge bus voltage is  ");
        PUT(MINIMUM_DISCHARGE_BUS_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);
        PUT("  volts");
        NEW_LINE(3);

    when 2=>
        VIDEO.CLEAR_SCREEN;
        NEW_LINE(2);
        PUT_LINE("*********************************************************************");
        NEW_LINE(2);
        PUT_LINE("Please enter the DESIGN bus voltage" );
        NEW_LINE(2);
        SET_COL(10);
        GET_DATA(BUS_VOLTAGE);
        VIDEO.CLEAR_SCREEN;
        PUT("DESIGN Bus voltage is  ");
        PUT(BUS_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);
        PUT("  volts");
        NEW_LINE(2);

    when 3=>
        VIDEO.CLEAR_SCREEN;
        NEW_LINE(2);
        PUT_LINE("*********************************************************************");
        NEW_LINE(2);
        PUT_LINE("Please enter the End of Life battery discharge voltage drop");
        PUT_LINE("about 1.0-2.0 volts");
        NEW_LINE(2);
        SET_COL(10);
        GET_DATA(BYPASS_DIODE_VOLTAGE_DROP);
        VIDEO.CLEAR_SCREEN;
        PUT("Minimum discharge bus voltage is  ");
        PUT(BYPASS_DIODE_VOLTAGE_DROP,FORE=>4,AFT=>2,EXP=>0);
        PUT("  volts");
        NEW_LINE(2);

    when 4=>
        VIDEO.CLEAR_SCREEN;
        NEW_LINE(2);
        PUT_LINE("*********************************************************************");
        NEW_LINE(2);
        SET_COL(5);
        PUT("Please enter End of life battery discharge voltage     [4] ");
        NEW_LINE(2);
        set_col(10);
        GET_DATA(EOL_BATTERY_DISCHARGE_VOLTAGE);
        VIDEO.CLEAR_SCREEN;
        NEW_LINE(2);
        PUT("End of life battery discharge voltage is [1.1 volts] ");
        PUT(EOL_BATTERY_DISCHARGE_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);
        PUT("  volts");
        NEW_LINE(2);
```

```
when 5=>
    VIDEO.CLEAR_SCREEN;
    NEW_LINE(2);
    PUT_LINE("*********************************************************************");
    NEW_LINE(2);
    PUT_LINE("Please enter the time satellite is in eclipse per orbit in hours");
    PUT_LINE("(about 1.2 hours in geosynchronous)");
    NEW_LINE(2);
    SET_COL(10);
    GET_DATA(ECLIPSE_TIME);
    VIDEO.CLEAR_SCREEN;
    PUT("Eclipse time is");
    PUT(ECLIPSE_TIME,FORE=>4,AFT=>2,EXP=>0);
    PUT("  hours");
    NEW_LINE(2);

when 6=>
    VIDEO.CLEAR_SCREEN;
    NEW_LINE(2);
    PUT_LINE("*********************************************************************");
    NEW_LINE(2);
    PUT("Please enter the Depth of Discharge used for batteries (0.50 - 0.75");
    NEW_LINE(2);
    SET_COL(10);
    GET_DATA(DEPTH_OF_DISCHARGE);
    VIDEO.CLEAR_SCREEN;
    PUT("Depth of discharge is  ");
    PUT(DEPTH_OF_DISCHARGE,FORE=>4,AFT=>2,EXP=>0);
    NEW_LINE(2);


 when 7=>
      VIDEO.CLEAR_SCREEN;
      NEW_LINE(2);
      PUT_LINE("*********************************************************************");
      NEW_LINE(2);
      PUT_LINE("Please enter the maximum battery discharge voltage ");
      PUT_LINE("(default 1.5 volts)");
      NEW_LINE(2);
      SET_COL(10);
      GET_DATA(MAXIMUM_BATTERY_CHARGE_VOLTAGE);
      NEW_LINE(1);
      SET_COL(5);
      PUT("Maximum Battery Discharge Voltage ");
      SET_COL(60);
      PUT(MAXIMUM_BATTERY_CHARGE_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);
      PUT("  volts");
      NEW_LINE(2);

 when 8=>
      VIDEO.CLEAR_SCREEN;
      NEW_LINE(2);
```

```
          PUT_LINE("**********************************************************************");
          NEW_LINE(2);
          PUT_LINE("Please enter the series connected diode voltage drop");
          PUT_LINE("(default 0.8 volts)");
          NEW_LINE(2);
          SET_COL(10);
          GET_DATA(SERIES_CONNECTED_DIODE_VOLTAGE_DROP);
          NEW_LINE(1);
          SET_COL(5);
          NEW_LINE(1);
          PUT("Series Connected Diode Voltage Drop ");
          SET_COL(60);
          PUT(SERIES_CONNECTED_DIODE_VOLTAGE_DROP,FORE=>4,AFT=>2,EXP=>0);
          PUT(" volts");
          NEW_LINE(2);


     when 9=>
          VIDEO.CLEAR_SCREEN;
          NEW_LINE(2);
          PUT_LINE("**********************************************************************");
          NEW_LINE(2);
          PUT_LINE("Please enter the number of series connected diodes");
          PUT_LINE("(default 3)");
          NEW_LINE(2);
          SET_COL(10);
          GET_DATA(NUMBER_SERIES_CONNECTED_DIODES);
          NEW_LINE(1);
          SET_COL(5);
          PUT("Number of Series Connected Diodes ");
          SET_COL(60);
          PUT(NUMBER_SERIES_CONNECTED_DIODES,FORE=>4,AFT=>2,EXP=>0);
          NEW_LINE(2);


     when 10=>
        VIDEO.CLEAR_SCREEN;
        NEW_LINE(2);
        PUT_LINE("**********************************************************************");
        NEW_LINE(2);
        PUT_LINE("Please enter the battery charger voltage drop");
        PUT_LINE("(default 1.75 volts)");
        NEW_LINE(2);
        SET_COL(10);
        GET_DATA(BATTERY_CHARGER_VOLTAGE_DROP);
        NEW_LINE(1);
        SET_COL(5);
        PUT("Battery Charger Voltage Drop ");
        SET_COL(60);
        PUT(BATTERY_CHARGER_VOLTAGE_DROP,FORE=>4,AFT=>2,EXP=>0);
        PUT(" volts");
        NEW_LINE(2);
```

```
  when 11=>
       VIDEO.CLEAR_SCREEN;
       NEW_LINE(2);
       PUT_LINE("**********************************************************************");
       NEW_LINE(2);
       PUT_LINE("Please enter the battery charge discharge efficiency");
       PUT_LINE("(default 0.9)");
       NEW_LINE(2);
       SET_COL(10);
       GET_DATA(CHARGE_DISCHARGE_EFFICIENCY_BATTERY);
       NEW_LINE(1);
       SET_COL(5);
       PUT("Charge Discharge Voltage Drop ");
       SET_COL(60);
       PUT(CHARGE_DISCHARGE_EFFICIENCY_BATTERY,FORE=>4,AFT=>2,EXP=>0);
       NEW_LINE(2);

  when OTHERS =>
       VIDEO.CLEAR_SCREEN;
       NEW_LINE(2);
       PUT_LINE("**********************************************************************");
       NEW_LINE(2);
       SET_COL(5);
       PUT_LINE("ENTER A PROPER NUMBER IDENTIFIER FOR VARIABLE TO CHANGE!");
       NEW_LINE(2);
  end case;

       NEW_LINE(2);
       PUT_LINE("************************************************************************");
       NEW_LINE(2);
       CHAR := N;
       PUT_LINE("If you wish to change a value please enter a  'y'  for YES");
       PUT_LINE("otherwise enter a 'n' for NO ");
       REPLACE:=FALSE;
       GET_CHARACTER(CHAR);
       if CHAR = 'Y' or CHAR = 'y' then
         CHAR := N;
         REPLACE:=TRUE;
         VIDEO.CLEAR_SCREEN;
         goto VALUE_NEW;

       else
         VIDEO.CLEAR_SCREEN;
         PUT_LINE("UNDERSTAND NO MORE CHANGES");
         NEW_LINE(3);
       end if;
  STOP;

<<KEEP_VALUES>>
  PUT_LINE("****************************************************************************");
  SET_COL(5);
  PUT_LINE("Values for the following parameters are: ");
PUT_LINE("*******************************************************************************");
SET_COL(5);
PUT("Minimum discharge voltage of bus");
set_col(60);
PUT(MINIMUM_DISCHARGE_BUS_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);PUT(" volts");

NEW_LINE(1);
SET_COL(5);
PUT("DESIGN Satellite Bus Voltage");
SET_COL(60);
PUT(BUS_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);
PUT(" volts");

NEW_LINE(1);
SET_COL(5);
PUT("Bypass diode voltage drop");
set_col(60);
PUT(BYPASS_DIODE_VOLTAGE_DROP,FORE=>4,AFT=>2,EXP=>0);PUT(" volts");

NEW_LINE(1);
SET_COL(5);
PUT("End of life battery discharge voltage");
set_col(60);
PUT(EOL_BATTERY_DISCHARGE_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);PUT(" volts");

NEW_LINE(1);
SET_COL(5);
PUT("Satellite eclipse time hours");
SET_COL(60);
PUT(ECLIPSE_TIME,FORE=>4,AFT=>2,EXP=>0);
PUT(" hours");

NEW_LINE(1);
SET_COL(5);
PUT("Depth of Discharge  ");
SET_COL(60);
PUT(DEPTH_OF_DISCHARGE,FORE=>4,AFT=>2,EXP=>0);

NEW_LINE(1);
SET_COL(5);
PUT("Maximum Battery Discharge Voltage ");
SET_COL(60);
PUT(MAXIMUM_BATTERY_CHARGE_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);
PUT(" volts");

NEW_LINE(1);
SET_COL(5);
PUT("Series Connected Diode Voltage Drop ");
SET_COL(60);
PUT(SERIES_CONNECTED_DIODE_VOLTAGE_DROP,FORE=>4,AFT=>2,EXP=>0);
PUT(" volts");
```

```
NEW_LINE(1);
SET_COL(5);
PUT("Number of Series Connected Diodes ");
SET_COL(60);
PUT(NUMBER_SERIES_CONNECTED_DIODES,FORE=>4,AFT=>2,EXP=>0);

NEW_LINE(1);
SET_COL(5);
PUT("Battery Charger Voltage Drop ");
SET_COL(60);
PUT(BATTERY_CHARGER_VOLTAGE_DROP,FORE=>4,AFT=>2,EXP=>0);
PUT("  volts");

NEW_LINE(1);
SET_COL(5);
PUT("Charge Discharge Voltage Drop ");
SET_COL(60);
PUT(CHARGE_DISCHARGE_EFFICIENCY_BATTERY,FORE=>4,AFT=>2,EXP=>0);
new_LINE(2);
STOP;

<<BUS>>
VIDEO.CLEAR_SCREEN;
NEW_LINE(2);
PUT_LINE("**********************************************************************");
NEW_LINE(2);
PUT_LINE("Please enter the number of buses used in your satellite.");
PUT_LINE("Most satellites have 2 buses.");
GET_INTEGER(J);
NEW_LINE(1);
PUT("Satellite has ");
PUT(J,WIDTH=>2);
PUT("  buses");
if J > 2 then
  NEW_LINE(1);
  PUT_LINE("You have selected more then two buses. Are you sure?");
  PUT_LINE("To change the number of buses enter a 'Y' for YES or 'N' for NO.");
  NEW_LINE(2);
  GET_CHARACTER(CHAR);
      if CHAR = 'Y' or CHAR = 'y' then
          CHAR := N;
          VIDEO.CLEAR_SCREEN;
          goto BUS;
      else
        VIDEO.CLEAR_SCREEN;
        PUT_LINE("UNDERSTAND MORE THAN TWO BUSES IS OKAY.");
        NEW_LINE(2);
      end if;
end if;

NEW_LINE(1);
VIDEO.CLEAR_SCREEN;
```

```
  PUT("Satellite has ");
  PUT(J,WIDTH=>2);
  PUT(" buses");
  NUMBER_OF_BUSES:=FLOAT(J);


end OPERATING_DATA;

procedure BATTERY  (BATTERY_LOAD               : in out FLOAT;
                    DRUM_SPINNER               : in out BOOLEAN;
                    MINIMUM_DISCHARGE_BUS_VOLTAGE : in out FLOAT;
                    BUS_VOLTAGE                : in out FLOAT;
                    BYPASS_DIODE_VOLTAGE_DROP  : in out FLOAT;
                    EOL_BATTERY_DISCHARGE_VOLTAGE : in out FLOAT;
                    BUS_POWER                  : in out FLOAT;
                    PAYLOAD_POWER              : in out FLOAT;
                    CELL_AH                    : in out FLOAT;
                    DEPTH_OF_DISCHARGE         : in out FLOAT;
                    ECLIPSE_TIME               : in out FLOAT;
                    VOLTAGE_CHARGE_ARRAY       : in out FLOAT;
                    NUMBER_OF_BUSES            : in out FLOAT;
                    MAXIMUM_BATTERY_CHARGE_VOLTAGE : in out FLOAT;
                    SERIES_CONNECTED_DIODE_VOLTAGE_DROP : in out FLOAT;
                    NUMBER_SERIES_CONNECTED_DIODES : in out FLOAT;
                    BATTERY_CHARGER_VOLTAGE_DROP : in out FLOAT;
                    CHARGE_DISCHARGE_EFFICIENCY_BATTERY : in out FLOAT;
                    spacecraft_life            : in out FLOAT;
                    POWER_EQUINOX_CHARGE       : in out FLOAT;
                    POWER_SOLSTICE_CHARGE      : in out FLOAT) is

 DESIGN_MARGIN                  : FLOAT := 1.1;
 SOLAR_INTENSITY                : FLOAT := 135.0;  -- mw / cm^2
 SOLAR_ARRAY_TEMP_SOLSTICE      : FLOAT := 37.0;   -- celcius
 SOLAR_ARRAY_TEMP_EQUINOX       : FLOAT := 45.0;   -- celcius
 SOLAR_CELL_TEST_TEMP           : FLOAT := 25.0;   -- celcius
 TEMP_COEF_EOL_CURRENT          : FLOAT := 0.00024; -- ma/cm^2
 TEMP_COEF_EOL_VOLTAGE          : FLOAT := -0.0022; -- ma/cm^2
 CURRENT_MAX_POWER              : FLOAT := 0.2966; -- A   Imp
 VOLTAGE_MAX_POWER              : FLOAT := 0.45;   -- V   Vmp
 CURRENT_SHORT_CIRCUIT          : FLOAT := 0.315;  -- A   Isc
 VOLTAGE_OPEN_CIRCUIT           : FLOAT := 0.548;  -- V   Voc
 CELL_WIDTH                     : FLOAT := 2.0;    -- cm
 CELL_LENGTH                    : FLOAT := 4.0;    -- cm
 CELL_THICKNESS                 : FLOAT := 0.021;  -- cm w/coverglass
 SOLSTICE_ARRAY_TEMP            : FLOAT := 37.0;   -- degrees celcius
 EQUINOX_ARRAY_TEMP             : FLOAT := 45.0;   -- degrees celcius
 ASSEMBLY_LOSS_CURRENT              : FLOAT:=0.9800;
 ENVIRONMENTAL_DEGRADATION_CURRENT  : FLOAT:=0.9077;
 ENVIRONMENTAL_DEGRADATION_VOLTAGE  : FLOAT:=0.9675;
-- SIF = SOLAR INTENSITY FACTOR
 SOLSTICE_SIF_CURRENT               : FLOAT:=0.94425;
 SOLSTICE_SIF_VOLTAGE               : FLOAT:=1.00000;
```

```
EQUINOX_SIF_CURRENT                    : FLOAT:=0.9970;
EQUINOX_S      OLTAGE                  : FLOAT:=1.0000;
EFF_ILLUMINATION_FLAT_PANEL                : FLOAT:=1.0000;
EFF_ILLUMINATION_SPIN                  : FLOAT:=3.14159265359;
PANEL_WIRING_LOSS_PER_CELL               : FLOAT:=0.005;    -- delta V
SOLSTICE_SOLAR_ARRAY_TEMP                : FLOAT:=39.0;  -- degree celcius
EQUINOX_SOLAR_ARRAY_TEMP                 : FLOAT:=49.0;  -- degree celcius
BLOCKING_DIODE_VOLTAGE_DROP               : FLOAT:=0.9;   --
ARRAY_WIRING_HARNESS_AND_SLIP_RING_VOLTAGE_DROP: FLOAT := 0.9;


BUS_CURRENT,
CELL_CURRENT_EOL_SOLSTICE,
CELL_CURRENT_EOL_EQUINOX,
CELL_VOLTAGE_EOL_SOLSTICE,
CELL_VOLTAGE_EOL_EQUINOX,

NUMBER_CELLS_IN_SERIES_CHARGE_ARRAY_SOLSTICE,
NUMBER_CELLS_IN_PARALLEL_CHARGE_ARRAY_SOLSTICE,
NUMBER_CELLS_IN_SERIES_CHARGE_ARRAY_EQUINOX,
NUMBER_CELLS_IN_PARALLEL_CHARGE_ARRAY_EQUINOX,

TOTAL_NUMBER_CELLS,
POWER_TOTAL,
TOTAL_POWER_PER_BUS,
ARRAY_TOTAL_POWER,
POWER_MARGIN,

REQUIRED_CURRENT_SOLSTICE_PER_BUS,
REQUIRED_CURRENT_EQUINOX_PER_BUS,
TEMP_OF_CELL_CHARACS,
CELL_AREA,
NUMBER_CELLS_IN_PARALLEL,
NUMBER_CELLS_IN_SERIES,
DESIGN_LOAD_SOLSTICE,
DESIGN_LOAD_EQUINOX                : FLOAT;

FINAL                         : BOOLEAN := TRUE;

PARAMETERS,
CELLS                         : INTEGER;

OUTSC                         : FILE_TYPE;




  EQUINOX_CURRENT,
  SOLSTICE_CURRENT,
  MAXIMUM_CHARGE_VOLTAGE,
  RECHARGE_TIME                : FLOAT ;
```

```
CHARGE                    : INTEGER ;

OUTF                     : FILE_TYPE;



begin
  NEW_LINE(2);
  PUT("Please enter the Solar Cell Test Temperature in degrees celcius");
  NEW_LINE(4);
  set_col(15);
  GET_DATA(SOLAR_CELL_TEST_TEMP);
  VIDEO.CLEAR_SCREEN;
  PUT("Solar Cell Test Temperature is ");
  PUT(SOLAR_CELL_TEST_TEMP,FORE=>3,AFT=>2,EXP=>0);
  PUT(" celcius");
  NEW_LINE(4);
  STOP;


  NEW_LINE(1);
  PUT_LINE("  The electrical characteristics of typical solar cells are based ");
  NEW_LINE(1);
  PUT("on ");
  PUT(SOLAR_CELL_TEST_TEMP,FORE=>2,AFT=>0,EXP=>0);
  PUT(" degrees celcius temperature and standard solar intensity ");
  NEW_LINE(2);
  PUT_LINE("of 135.3 mw/cm^2 on a cell basis at BOL.  The realistic solar");
  NEW_LINE(1);
  PUT_LINE("array power is, however, calculated by considering several  ");
  NEW_LINE(1);
  PUT_LINE("factors, such as assembly loss factors, environmental degradation,");
  NEW_LINE(1);
  PUT_LINE("the seasonal variation of solar intensity, solar cell temperature, ");
  NEW_LINE(1);
  PUT_LINE("and random failures. Some typical values are given in Table 6.6 ");
  NEW_LINE(1);
  PUT_LINE("of Brij N. Agrawal's book Design of Geosynchronous Spacecraft.");
  NEW_LINE(2);
  STOP;

  if SPACECRAFT_LIFE = 10.01 then
    ASSEMBLY_LOSS_CURRENT               :=0.9600;
    ENVIRONMENTAL_DEGRADATION_CURRENT   :=0.8561;
    ENVIRONMENTAL_DEGRADATION_VOLTAGE   :=0.9350;
    -- SIF = SOLAR INTENSITY FACTOR
    SOLSTICE_SIF_CURRENT              :=0.8885;
    SOLSTICE_SIF_VOLTAGE              :=1.00000;
    EQUINOX_SIF_CURRENT               :=0.9941;
    EQUINOX_SIF_VOLTAGE               :=1.0000;
```

```
elsif SPACECRAFT_LIFE <= 2.0 then
  ASSEMBLY_LOSS_CURRENT              :=0.9800;
  ENVIRONMENTAL_DEGRADATION_CURRENT   :=0.9077;
  ENVIRONMENTAL_DEGRADATION_VOLTAGE   :=0.9675;
  -- SIF = SOLAR INTENSITY FACTOR
  SOLSTICE_SIF_CURRENT              :=0.94425;
  SOLSTICE_SIF_VOLTAGE              :=1.00000;
  EQUINOX_SIF_CURRENT               :=0.9970;
  EQUINOX_SIF_VOLTAGE               :=1.0000;


elsif SPACECRAFT_LIFE > 2.0 and SPACECRAFT_LIFE < 7.0 then

  ASSEMBLY_LOSS_CURRENT:=ASSEMBLY_LOSS_CURRENT
                    /(1.011722**INTEGER(SPACECRAFT_LIFE-2.0));

  ENVIRONMENTAL_DEGRADATION_CURRENT:=ENVIRONMENTAL_DEGRADATION_CURRENT
                                /1.0216787**INTEGER(SPACECRAFT_LIFE-2.0);

  ENVIRONMENTAL_DEGRADATION_VOLTAGE:=ENVIRONMENTAL_DEGRADATION_VOLTAGE
                                /1.0068572**INTEGER(SPACECRAFT_LIFE-2.0);
  -- SIF = SOLAR INTENSITY FACTOR

  SOLSTICE_SIF_CURRENT:=SOLSTICE_SIF_CURRENT
                    /1.0111722**INTEGER(SPACECRAFT_LIFE-2.0);

  EQUINOX_SIF_CURRENT:=EQUINOX_SIF_CURRENT
                    /1.000582763**INTEGER(SPACECRAFT_LIFE-2.0);


elsif SPACECRAFT_LIFE >= 7.0 and SPACECRAFT_LIFE < 8.0 then

  ASSEMBLY_LOSS_CURRENT                := 0.96;
  ENVIRONMENTAL_DEGRADATION_CURRENT      := 0.8154;
  ENVIRONMENTAL_DEGRADATION_VOLTAGE      := 0.935;
  -- SIF = SOLAR INTENSITY FACTOR
  SOLSTICE_SIF_CURRENT               := 0.8885;
  SOLSTICE_SIF_VOLTAGE               := 1.0;
  EQUINOX_SIF_CURRENT                := 0.9941;
  EQUINOX_SIF_VOLTAGE                := 1.0;
  EFF_ILLUMINATION_FLAT_PANEL           := 1.0;
  EFF_ILLUMINATION_SPIN              := 1.0;

elsif SPACECRAFT_LIFE >= 8.0 and SPACECRAFT_LIFE <= 50.0 then
  -- 1.016397 is a mathematical constant to cause a continuous
  -- 5% degradation every 5 years.

  ASSEMBLY_LOSS_CURRENT:=ASSEMBLY_LOSS_CURRENT
                    /1.016397**INTEGER(SPACECRAFT_LIFE-7.0);
```

```
    ENVIRONMENTAL_DEGRADATION_CURRENT:=ENVIRONMENTAL_DEGRADATION_CURRENT
                                  /1.016397**INTEGER(SPACECRAFT_LIFE-7.0);

    ENVIRONMENTAL_DEGRADATION_VOLTAGE:=ENVIRONMENTAL_DEGRADATION_VOLTAGE
                                  /1.016397**INTEGER(SPACECRAFT_LIFE-7.0);
    -- SIF = SOLAR INTENSITY FACTOR

    SOLSTICE_SIF_CURRENT:=SOLSTICE_SIF_CURRENT
                         /1.016397**INTEGER(SPACECRAFT_LIFE-7.0);

    EQUINOX_SIF_CURRENT:=EQUINOX_SIF_CURRENT
                        /1.016397**INTEGER(SPACECRAFT_LIFE-7.0);

  else
    VIDEO.CLEAR_SCREEN;
    PUT_LINE(" Spacecraft life to long   DEFAULT values will be used");

  end if;

  PUT_LINE("Environmental Degradation Factors for given Spacecraft Life");
  PUT_LINE("*************************************************************");
  new_line(1);

  PUT("Assembly Loss Current is ");
  SET_COL(60);
  PUT(ASSEMBLY_LOSS_CURRENT,FORE=>1,AFT=>4,EXP=>0);
  NEW_LINE(2);

  PUT("Environmental Degradation Current is ");
  SET_COL(60);
  PUT(ENVIRONMENTAL_DEGRADATION_CURRENT,FORE=>1,AFT=>4,EXP=>0);
  NEW_LINE(2);

  PUT("Environmental Degradation Voltage is ");
  SET_COL(60);
  PUT(ENVIRONMENTAL_DEGRADATION_VOLTAGE,FORE=>1,AFT=>4,EXP=>0);
  NEW_LINE(2);

  PUT("Solar Intensity Factor for Current during Solstice  is ");
  SET_COL(60);
  PUT(SOLSTICE_SIF_CURRENT,FORE=>1,AFT=>4,EXP=>0);
  NEW_LINE(2);

  PUT("Solar Intensity Factor for Voltage during Solstice  is ");
  SET_COL(60);
  PUT(SOLSTICE_SIF_VOLTAGE,FORE=>1,AFT=>4,EXP=>0);
  NEW_LINE(2);

  PUT("Solar Intensity Factor for Current during Equinox is ");
  SET_COL(60);
  PUT(EQUINOX_SIF_CURRENT,FORE=>1,AFT=>4,EXP=>0);
  NEW_LINE(2);
```

```
PUT("Solar Intensity Factor for Voltage during Equinox is ");
SET_COL(60);
PUT(EQUINOX_SIF_VOLTAGE,FORE=>1,AFT=>4,EXP=>0);
NEW_LINE(2);



NEW_LINE(2);
PUT_LINE("To use the above calculated values enter a    '1'");
NEW_LINE(1);
PUT_LINE("To change the calculated    values enter a    '2'");
NEW_LINE(1);

SET_COL(5);
GET_INTEGER(CELLS);

case CELLS is
  when 1=>
      PUT_LINE("Understand default values will be used");

  when 2=>
      VIDEO.CLEAR_SCREEN;
      new_line(2);
      PUT_LINE("**********************************************************************");
      NEW_LINE(2);
      PUT("Please enter desired value for Assembly Loss Current ");
      NEW_LINE(2);
      SET_COL(15);
      GET_DATA(ASSEMBLY_LOSS_CURRENT);

      VIDEO.CLEAR_SCREEN;
      new_line(2);
      PUT_LINE("**********************************************************************");
      NEW_LINE(2);
      PUT("Please enter desired value for Environmental Degradation (current)");
      NEW_LINE(2);
      SET_COL(15);
      GET_DATA(ENVIRONMENTAL_DEGRADATION_CURRENT);

      VIDEO.CLEAR_SCREEN;
      new_line(2);
      PUT_LINE("**********************************************************************");
      NEW_LINE(2);
      PUT("Please enter desired value for Environmental Degradation (voltage)");
      NEW_LINE(2);
      SET_COL(15);
      GET_DATA(ENVIRONMENTAL_DEGRADATION_VOLTAGE);

      VIDEO.CLEAR_SCREEN;
      new_line(2);
      PUT_LINE("**********************************************************************");
      NEW_LINE(2);
```

```
        PUT_LINE("Please enter desired value for ");
        PUT_LINE("   Solar Intensity Factor (solstice current)");
        NEW_LINE(2);
        SET_COL(15);
        GET_DATA(SOLSTICE_SIF_CURRENT);

        VIDEO.CLEAR_SCREEN;
        new_line(2);
        PUT_LINE("******************************************************************");
        NEW_LINE(2);
        PUT_LINE("Please enter desired value for ");
        PUT_LINE("   Solar Intensity Factor (solstice voltage)");
        NEW_LINE(2);
        SET_COL(15);
        GET_DATA(SOLSTICE_SIF_VOLTAGE);

        VIDEO.CLEAR_SCREEN;
        new_line(2);
        PUT_LINE("******************************************************************");
        NEW_LINE(2);
        PUT_LINE("Please enter desired value for ");
        PUT_LINE("   Solar Intensity Factor (equinox current)");
        NEW_LINE(2);
        SET_COL(15);
        GET_DATA(EQUINOX_SIF_CURRENT);

        VIDEO.CLEAR_SCREEN;
        new_line(2);
        PUT_LINE("******************************************************************");
        NEW_LINE(2);
        PUT_LINE("Please enter desired value for ");
        PUT_LINE("   Solar Intensity Factor (equinox voltage)");
        NEW_LINE(2);
        SET_COL(15);
        GET_DATA(EQUINOX_SIF_VOLTAGE);

    when OTHERS =>
        VIDEO.CLEAR_SCREEN;
        NEW_LINE(2);
        SET_COL(5);
        PUT_LINE("Understand Calculated values will be used");
        NEW_LINE(2);
  end case; -- CHARGE
  VIDEO.CLEAR_SCREEN;
  NEW_LINE(1);
  PUT("SOLAR ARRAY DESIGN FACTORS ARE:");
  NEW_LINE(2);
  PUT("Assembly Loss Current is ");
  SET_COL(60);
  PUT(ASSEMBLY_LOSS_CURRENT,FORE=>1,AFT=>4,EXP=>0);
  NEW_LINE(2);
```

```
PUT("Environmental Degradation Current is ");
SET_COL(60);
PUT(ENVIRONMENTAL_DEGRADATION_CURRENT,FORE=>1,AFT=>4,EXP=>0);
NEW_LINE(2);

PUT("Environmental Degradation Voltage is ");
SET_COL(60);
PUT(ENVIRONMENTAL_DEGRADATION_VOLTAGE,FORE=>1,AFT=>4,EXP=>0);
NEW_LINE(2);

PUT("Solar Intensity Factor for Current during Solstice  is ");
SET_COL(60);
PUT(SOLSTICE_SIF_CURRENT,FORE=>1,AFT=>4,EXP=>0);
NEW_LINE(2);

PUT("Solar Intensity Factor for Voltage during Solstice  is ");
SET_COL(60);
PUT(SOLSTICE_SIF_VOLTAGE,FORE=>1,AFT=>4,EXP=>0);
NEW_LINE(2);

PUT("Solar Intensity Factor for Current during Equinox is ");
SET_COL(60);
PUT(EQUINOX_SIF_CURRENT,FORE=>1,AFT=>4,EXP=>0);
NEW_LINE(2);

PUT("Solar Intensity Factor for Voltage during Equinox is ");
SET_COL(60);
PUT(EQUINOX_SIF_VOLTAGE,FORE=>1,AFT=>4,EXP=>0);
NEW_LINE(3);
STOP;
NEW_LINE(1);
PUT_LINE("******************************************************************");
NEW_LINE(2);
PUT_LINE("                    SOLAR CELL CHARACTERISTICS");
PUT_LINE("******************************************************************");
PUT_LINE("Choice        '1'      '2'      '3'  '4'     '5'      '6'       ");
PUT_LINE("Charac     Intelsat Intelsat  Intelsat  GaAs/Ge   AlAs/Ge ");
PUT_LINE("           IV        V         VI                        ");
PUT_LINE("           Si        Si      ------------                ");
PUT_LINE("                          K4-3/4   K7 ");
PUT_LINE("------------------------------------------------------------ ");

PUT_LINE("Imp  Amps   0.125   0.2966   0.391   0.644   0.725   0.780");

PUT_LINE("Vmp  Volts  0.445   0.450    0.454   0.478   0.885   0.812");

PUT_LINE("Isc  Amps   0.141   0.315    0.4187  0.6887  0.735   0.878");

PUT_LINE("Voc  Volts  0.560   0.548    0.545   0.590   0.889   0.851");

PUT_LINE("Size cm     2x2   2.1x4.0  1.8x6.2 2.5x6.2   2x4     2x2 ");
PUT_LINE("Thickness   0.033  0.025   0.020  0.020  .020    0.20");

PUT_LINE("Material      Si      Si      Si      Si    GaAs/Ge  AlAs/Ge");

PUT_LINE("To use the above calculated values enter the appropriate");

PUT_LINE("number located at the top of the column.  To enter your own");

PUT_LINE("cell parameters enter a '7'");

SET_COL(5);
GET_INTEGER(PARAMETERS);

case PARAMETERS is
  when 1=>
      TEMP_COEF_EOL_CURRENT         := 0.00024;  -- ma/cm^2
      TEMP_COEF_EOL_VOLTAGE         := -0.0022;
      CURRENT_MAX_POWER            := 0.1250;   -- A   Imp
      VOLTAGE_MAX_POWER            := 0.445;    -- V   Vmp
      CURRENT_SHORT_CIRCUIT        := 0.141;    -- A   Isc
      VOLTAGE_OPEN_CIRCUIT         := 0.560;    -- V   Voc
      CELL_WIDTH            := 2.0;     -- cm
      CELL_LENGTH           := 2.0;     -- cm
      CELL_THICKNESS         := 0.033;   -- cm

  when 2=>
      TEMP_COEF_EOL_CURRENT         := 0.00024;  -- ma/cm^2
      TEMP_COEF_EOL_VOLTAGE         := -0.0022;
      CURRENT_MAX_POWER            := 0.2966;   -- A   Imp
      VOLTAGE_MAX_POWER            := 0.45;     -- V   Vmp
      CURRENT_SHORT_CIRCUIT        := 0.315;    -- A   Isc
      VOLTAGE_OPEN_CIRCUIT         := 0.548;    -- V   Voc
      CELL_WIDTH            := 2.0;     -- cm
      CELL_LENGTH           := 4.0;     -- cm
      CELL_THICKNESS         := 0.025;   -- cm

  when 3=>
      TEMP_COEF_EOL_CURRENT         := 0.00024;  -- ma/cm^2
      TEMP_COEF_EOL_VOLTAGE         := -0.0022;
      CURRENT_MAX_POWER            := 0.391;    -- A   Imp
      VOLTAGE_MAX_POWER            := 0.454;    -- V   Vmp
      CURRENT_SHORT_CIRCUIT        := 0.4187;   -- A   Isc
      VOLTAGE_OPEN_CIRCUIT         := 0.545;    -- V   Voc
      CELL_WIDTH            := 1.8;     -- cm
      CELL_LENGTH           := 6.2;     -- cm
      CELL_THICKNESS         := 0.02;    -- cm

  when 4=>
      TEMP_COEF_EOL_CURRENT         := 0.00024;  -- ma/cm^2
      TEMP_COEF_EOL_VOLTAGE         := -0.0022;
      CURRENT_MAX_POWER            := 0.644;    -- A   Imp
      VOLTAGE_MAX_POWER            := 0.478;    -- V   Vmp
```

```
    CURRENT_SHORT_CIRCUIT          := 0.6887;   -- A   Isc
    VOLTAGE_OPEN_CIRCUIT           := 0.590;    -- V   Voc
    CELL_WIDTH                 := 2.085;     -- cm
    CELL_LENGTH                := 6.205;     -- cm
    CELL_THICKNESS              := 0.02;     -- cm

when 5=>
    TEMP_COEF_EOL_CURRENT          := 0.00024;  -- ma/cm^2
    TEMP_COEF_EOL_VOLTAGE          := -0.0022;
    CURRENT_MAX_POWER              := 0.725;    -- A   Imp
    VOLTAGE_MAX_POWER              := 0.750;    -- V   Vmp
    CURRENT_SHORT_CIRCUIT          := 0.735;    -- A   Isc
    VOLTAGE_OPEN_CIRCUIT           := 0.888;    -- V   Voc
    CELL_WIDTH                 := 2.0;       -- cm
    CELL_LENGTH                := 4.0;       -- cm
    CELL_THICKNESS              := 0.02;     -- cm


when 6=>
    TEMP_COEF_EOL_CURRENT          := 0.00024;  -- ma/cm^2
    TEMP_COEF_EOL_VOLTAGE          := -0.0022;
    CURRENT_MAX_POWER              := 0.780;    -- A   Imp
    VOLTAGE_MAX_POWER              := 0.812;    -- V   Vmp
    CURRENT_SHORT_CIRCUIT          := 0.878;    -- A   Isc
    VOLTAGE_OPEN_CIRCUIT           := 0.851;    -- V   Voc
    CELL_WIDTH                 := 2.0;       -- cm
    CELL_LENGTH                := 2.0;       -- cm
    CELL_THICKNESS              := 0.02;     -- cm



when 7 =>
    VIDEO.CLEAR_SCREEN;
    PUT_LINE("*************************************************************");
    PUT_LINE("Please enter the value for temperature coefficient EOL CURRENT");
    PUT_LINE("Default value is  -0.0022 A / degree C ");
    SET_COL(15);
    GET_DATA(TEMP_COEF_EOL_CURRENT);
    NEW_LINE(2);
    VIDEO.CLEAR_SCREEN;
    SET_COL(15);
    PUT("Temperature coefficient EOL CURRENT is ");
    PUT(TEMP_COEF_EOL_CURRENT,FORE=>1,AFT=>4,EXP=>0);
    PUT("  ma/cm^2"); NEW_LINE(2);

    PUT_LINE("*************************************************************");
    PUT_LINE("Please enter the value for temperature coefficient EOL VOLTAGE");
    PUT_LINE("Default value is  0.00024 V / degree C");
    SET_COL(15);
    GET_DATA(TEMP_COEF_EOL_VOLTAGE);
    VIDEO.CLEAR_SCREEN;
```

```
NEW_LINE(2);
SET_COL(15);
PUT("Temperature coefficient EOL VOLTAGE is ");
PUT(TEMP_COEF_EOL_VOLTAGE,FORE=>1,AFT=>4,EXP=>0);
PUT(" mv/cm^2"); NEW_LINE(2);

PUT_LINE("****************************************************************");
PUT_LINE("Please enter the value for CURRENT at MAXIMUM POWER (Imp) ");

PUT_LINE("Default value is 0.2966 Amps");
SET_COL(15);
GET_DATA(CURRENT_MAX_POWER );
VIDEO.CLEAR_SCREEN;
NEW_LINE(2);
SET_COL(15);
PUT("Current at Maximum Power is ");
PUT(CURRENT_MAX_POWER ,FORE=>1,AFT=>4,EXP=>0);
PUT(" Amps   -- Imp"); NEW_LINE(2);

PUT_LINE("****************************************************************");
PUT_LINE("Please enter the value for VOLTAGE at MAXIMUM POWER (Imp) ");
PUT_LINE("Default value is 0.45 Volts ");
SET_COL(15);
GET_DATA(VOLTAGE_MAX_POWER );
VIDEO.CLEAR_SCREEN;
NEW_LINE(2);
SET_COL(15);
FUT("Voltage at Maximum Power is ");
PUT(VOLTAGE_MAX_POWER ,FORE=>1,AFT=>4,EXP=>0);
PUT(" Volts   -- Vmp"); NEW_LINE(2);

PUT_LINE("****************************************************************");
PUT_LINE("Please enter the value for Short Circuit Current (Isc) ");
PUT_LINE("Default value is 0.315 Amps");
SET_COL(15);
GET_DATA(CURRENT_SHORT_CIRCUIT  );
VIDEO.CLEAR_SCREEN;
NEW_LINE(2);
SET_COL(15);
PUT("Short Circuit Current is ");
PUT(CURRENT_SHORT_CIRCUIT  ,FORE=>1,AFT=>4,EXP=>0);
PUT(" Amps   -- Isc");NEW_LINE(2);

PUT_LINE("****************************************************************");
PUT_LINE("Please enter the value for Open Circuit Voltage (Voc) ");
PUT_LINE("Default value is 0.548 Volts ");
SET_COL(15);
GET_DATA(VOLTAGE_OPEN_CIRCUIT );
VIDEO.CLEAR_SCREEN;
NEW_LINE(2);
SET_COL(15);
PUT("Open Circuit Voltage is ");
```

```
        PUT(VOLTAGE_OPEN_CIRCUIT  ,FORE=>1,AFT=>4,EXP=>0);
        PUT("  Volts    -- Voc");NEW_LINE(2);

        PUT_LINE("**************************************************************");
        PUT_LINE("Please enter the value for Solar Cell Width in cm.");
        PUT_LINE("Default value is  2 cm ");
        SET_COL(15);
        GET_DATA(CELL_WIDTH );
        VIDEO.CLEAR_SCREEN;
        NEW_LINE(2);
        SET_COL(15);
        PUT("Solar Cell Width is ");
        PUT(CELL_WIDTH,FORE=>2,AFT=>2,EXP=>0);
        PUT("  cm");NEW_LINE(2);

        PUT_LINE("**************************************************************");
        PUT("Please enter the value for Solar Cell Length in cm.");
        PUT_LINE("Default value is  4 cm");
        SET_COL(15);
        GET_DATA(CELL_LENGTH );
        VIDEO.CLEAR_SCREEN;
        NEW_LINE(2);
        SET_COL(15);
        PUT("Solar Cell Length is ");
        PUT(CELL_LENGTH,FORE=>2,AFT=>2,EXP=>0);
        PUT("  cm"); NEW_LINE(2);

        PUT_LINE("**************************************************************");
        PUT("Please enter the value for Solar Cell Thickness in cm. ");
        PUT_LINE("Default value is 0.025 cm ");
        SET_COL(15);
        GET_DATA(CELL_THICKNESS);
        VIDEO.CLEAR_SCREEN;

        NEW_LINE(2);
        SET_COL(15);
        PUT("Solar Cell Thickness is ");
        PUT(CELL_THICKNESS,FORE=>2,AFT=>4,EXP=>0);
        PUT("  cm");NEW_LINE(2);

    when OTHERS =>
        VIDEO.CLEAR_SCREEN;
        NEW_LINE(2);
        SET_COL(5);
        PUT_LINE("Understand Default values will be used");

  end case;  -- CHARGE
 -- STOP;
   VIDEO.CLEAR_SCREEN;
   PUT_LINE("**********************************************************");
        PUT_LINE("            SOLAR CELL PARAMETERS ARE   ");
        NEW_LINE(2);
```

```
SET_COL(15);
PUT("Temperature coefficient EOL CURRENT is ");
SET_COL(60);
PUT(TEMP_COEF_EOL_CURRENT,FORE=>1,AFT=>4,EXP=>0);
PUT("  ma/cm^2");

NEW_LINE(1);
SET_COL(15);
PUT("Temperature coefficient EOL VOLTAGE is ");
SET_COL(59);
PUT(TEMP_COEF_EOL_VOLTAGE,FORE=>1,AFT=>4,EXP=>0);
PUT("  ma/cm^2");

NEW_LINE(1);
SET_COL(15);
PUT("Current at Maximum Power is ");
SET_COL(60);
PUT(CURRENT_MAX_POWER ,FORE=>1,AFT=>4,EXP=>0);
PUT("  Amps (Imp)");

NEW_LINE(1);
SET_COL(15);
PUT("Voltage at Maximum Power is ");
SET_COL(60);
PUT(VOLTAGE_MAX_POWER ,FORE=>1,AFT=>4,EXP=>0);
PUT("  Volts (Vmp)");

NEW_LINE(1);
SET_COL(15);
PUT("Short Circuit Current is ");
SET_COL(60);
PUT(CURRENT_SHORT_CIRCUIT  ,FORE=>1,AFT=>4,EXP=>0);
PUT("  Amps (Isc)");

NEW_LINE(1);
SET_COL(15);
PUT("Open Circuit Voltage is ");
SET_col(60);
PUT(VOLTAGE_OPEN_CIRCUIT  ,FORE=>1,AFT=>4,EXP=>0);
PUT("  Volts (Voc)");

NEW_LINE(1);
SET_COL(15);
PUT("Solar Cell Width is ");
SET_COL(59);
PUT(CELL_WIDTH,FORE=>2,AFT=>4,EXP=>0);
PUT("  cm");

NEW_LINE(1);
SET_COL(15);
PUT("Solar Cell Length is ");
SET_COL(59);
```

```
        PUT(CELL_LENGTH,FORE=>2,AFT=>4,EXP=>0);
        PUT(" cm");

        NEW_LINE(1);
        SET_COL(15);
        PUT("Solar Cell Thickness is ");
        SET_COL(59);
        PUT(CELL_THICKNESS,FORE=>2,AFT=>4,EXP=>0);
        PUT(" cm");
        NEW_LINE(1);

        PUT_LINE("**************************************************************");
        NEW_LINE(3);
        STOP;
        -- Cell current at maximum power point of EOL summer solstice

-- PUT(CURRENT_MAX_POWER,FORE=>4,AFT=>4,EXP=>0);
-- NEW_LINE(1);
-- PUT(SOLSTICE_SOLAR_ARRAY_TEMP   ,FORE=>4,AFT=>4,EXP=>0);
-- NEW_LINE(1);
-- PUT( TEMP_COEF_EOL_CURRENT,FORE=>4,AFT=>4,EXP=>0);
-- NEW_LINE(1);
-- PUT(SOLAR_CELL_TEST_TEMP,FORE=>4,AFT=>4,EXP=>0);
-- NEW_LINE(1);
-- PUT(ENVIRONMENTAL_DEGRADATION_CURRENT   ,FORE=>4,AFT=>4,EXP=>0);
-- NEW_LINE(1);
-- PUT(ASSEMBLY_LOSS_CURRENT  ,FORE=>4,AFT=>4,EXP=>0);
-- NEW_LINE(1);




<<REITERATE_VOLTAGE>>


  if FINAL = FALSE then
    VIDEO.CLEAR_SCREEN;
    FINAL:=TRUE;
    PUT_LINE("PREVIOUSLY ENTERED DEFAULT VALUES OF FIRST ITERATION WILL BE USED");
    PUT_LINE("BE USED ON THIS ITERATION EXCEPT FOR REQUESTED VALUES");
    PUT_LINE("******************************************************************");


    NEW_LINE(2);
    PUT_LINE("Please enter the minimum satellite bus discharge voltage");
    PUT_LINE("about 30 volts");
    SET_COL(10);
    GET_DATA(MINIMUM_DISCHARGE_BUS_VOLTAGE);
    VIDEO.CLEAR_SCREEN;
    PUT("Minimum discharge bus voltage is  ");
  PUT(MINIMUM_DISCHARGE_BUS_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);
  PUT("  volts");

  NEW_LINE(2);
  PUT_LINE("**********************************************     ***************");
  NEW_LINE(2);
  PUT("Please enter the DESIGN satellite bus voltage");
  SET_COL(10);
  GET_DATA(BUS_VOLTAGE);
  VIDEO.CLEAR_SCREEN;
  PUT("Bus voltage is  ");
  PUT(BUS_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);
  PUT("  volts");
end if;


NEW_LINE(2);
SET_COL(1);
PUT("Series cells required for minimum discharge voltage FLOAT");
SERIES_CELLS_FOR_MIN_DIS_VOLT:=
        ((MINIMUM_DISCHARGE_BUS_VOLTAGE+BYPASS_DIODE_VOLTAGE_DROP)
        /EOL_BATTERY_DISCHARGE_VOLTAGE)+1.0;   --
SET_COL(64);
PUT(SERIES_CELLS_FOR_MIN_DIS_VOLT, FORE => 4, AFT => 2, EXP => 0);

N_INTEGER:=INTEGER(SERIES_CELLS_FOR_MIN_DIS_VOLT);
SET_COL(1);
PUT("Series cells required for minimum discharge voltage INTEGER");
SET_COL(64);
PUT(FLOAT(N_INTEGER),FORE=>4,AFT=>2,EXP=>0);
if FLOAT(N_INTEGER) < SERIES_CELLS_FOR_MIN_DIS_VOLT  then
  N_INTEGER:=N_INTEGER+1;
end if;
SERIES_CELLS_FOR_MIN_DIS_VOLT:=FLOAT(N_INTEGER);
SET_COL(1);
PUT("Series cells required for minimum discharge voltage  ROUND UP");
SET_COL(64);
PUT(SERIES_CELLS_FOR_MIN_DIS_VOLT,FORE=>4,AFT=>2,EXP=>0);
NEW_LINE(2);

PUT("Minimum discharge voltage is ");
MINIMUM_DISCHARGE_BUS_VOLTAGE:=((SERIES_CELLS_FOR_MIN_DIS_VOLT-1.0)
              *EOL_BATTERY_DISCHARGE_VOLTAGE)-BYPASS_DIODE_VOLTAGE_DROP;
SET_COL(64);
PUT(MINIMUM_DISCHARGE_BUS_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);PUT("  volts");
new_line(4);

STOP;

new_line(2);
PUT_LINE("  The power required by a bus will be all or part of the total");
PUT_LINE("power.  Therefore the  batteries  for  each  bus  must  provide");
PUT_LINE("power for the maximum  eclipse time of  1.2  hours for");
PUT_LINE("geosynchronous orbits.");
new_line(2);

CELL_AH:=(BATTERY_LOAD/NUMBER_OF_BUSES*ECLIPSE_TIME)
          /(MINIMUM_DISCHARGE_BUS_VOLTAGE*DEPTH_OF_DISCHARGE);

PUT("  The required Cell capacity is ");
PUT(CELL_AH,FORE=>3,AFT=>2,EXP=>0);
PUT(" ampere hours");

NEW_LINE(2);
BUS_POWER:=BATTERY_LOAD/NUMBER_OF_BUSES;
PUT("Bus Power is ");
PUT(BUS_POWER,FORE=>4,AFT=>2,EXP=>0);PUT("  watts");
new_line(2);
PUT_LINE("  To determine the maximum charge voltage it is assumed that an");
PUT("open circuit failure of a battery cell during charge is  ");
PUT(SERIES_CONNECTED_DIODE_VOLTAGE_DROP,FORE=>1,AFT=>2,EXP=>0);
PUT("  volts");
NEW_LINE(1);
PUT_LINE("per diode usually accomodated by three series connected ");
PUT_LINE("silicon diodes connected in parallel with the cell.");
NEW_LINE(2);
STOP;


PUT_LINE("*********************************************************************");
NEW_LINE(2);

MAXIMUM_CHARGE_VOLTAGE:=MAXIMUM_BATTERY_CHARGE_VOLTAGE
                        *(SERIES_CELLS_FOR_MIN_DIS_VOLT-1.0)
                        +NUMBER_SERIES_CONNECTED_DIODES
                        *SERIES_CONNECTED_DIODE_VOLTAGE_DROP;
VIDEO.CLEAR_SCREEN;
PUT("Maximum Charge Voltage is");
SET_COL(64);
PUT(MAXIMUM_CHARGE_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);
PUT("  volts");
NEW_LINE(2);

STOP;
NEW_LINE(2);
PUT("  The main bus voltage is regulated to + or - ");
PUT(BUS_VOLTAGE_ALLOWABLE_DEVIATION,FORE=>1,AFT=>2,EXP=>0);
PUT("  volts");
NEW_LINE(1);
PUT("Therefore the lower limit of the bus voltage is  ");
PUT(BUS_VOLTAGE-BUS_VOLTAGE_ALLOWABLE_DEVIATION,FORE=>2,AFT=>2,EXP=>0);
PUT("  volts");
NEW_LINE(1);
PUT("The battery charger voltage drop (V_cd) is ");
```

```
  PUT(BATTERY_CHARGER_VOLTAGE_DROP,FORE=>2,AFT=>2,EXP=>0);
  PUT(" volts");
  NEW_LINE(1);

VOLTAGE_CHARGE_ARRAY:=MAXIMUM_CHARGE_VOLTAGE-(BUS_VOLTAGE-BUS_VOLTAGE_A
LLOWABLE_DEVIATION)
                        +BATTERY_CHARGER_VOLTAGE_DROP;

  new_line(2);
  SET_COL(5);
  PUT("Then the boost voltage needed by the charge array is  ");
  PUT(VOLTAGE_CHARGE_ARRAY, FORE=>2,AFT=>2,EXP=>0);
  PUT(" volts");
  NEW_LINE(1);
  PUT_LINE("   The charge current is applied to each bus on a 50% duty");
  PUT_LINE("cycle.  The charge rates are CELL ampere hours/15 for autumnal ");
  PUT_LINE("equinox and CELL ampere hours/45 for summer solstice.  ");
  new_line(3);
  STOP;
  new_line(1);
  PUT_LINE("The currents are:");
  NEW_LINE(1);

  EQUINOX_CURRENT:=CELL_AH/EQUINOX_CHARGE_RATE;
  PUT("Equinox current is ");
  SET_COL(64);
  PUT(EQUINOX_CURRENT,FORE=>2,AFT=>2,EXP=>0);
  PUT(" amps");
  NEW_LINE(2);

  SOLSTICE_CURRENT:=CELL_AH/SOLSTICE_CHARGE_RATE;
  PUT("Solstice current is ");
  SET_COL(64);
  PUT(SOLSTICE_CURRENT,FORE=>2,AFT=>2,EXP=>0);
  PUT(" amps");
  NEW_LINE(2);

  PUT_LINE("Charging or discharging at high rate is based on returning");
  PUT_LINE("the energy depleted during eclipse to each battery.");
  NEW_LINE(1);
  POWER_EQUINOX_CHARGE:=MAXIMUM_CHARGE_VOLTAGE*EQUINOX_CURRENT;
  PUT("Power required for equinox charge at high rate is ");
  SET_COL(64);
  PUT(POWER_EQUINOX_CHARGE,FORE=>3,AFT=>2,EXP=>0);
  PUT(" watts");
  NEW_LINE(2);

  POWER_SOLSTICE_CHARGE:=MAXIMUM_CHARGE_VOLTAGE*SOLSTICE_CURRENT;
  PUT("Power required for solstice charge at high rate is ");
  SET_COL(64);
  PUT(POWER_SOLSTICE_CHARGE,FORE=>3,AFT=>2,EXP=>0);
  PUT(" watts");
```

```
NEW_LINE(2);

RECHARGE_TIME:=((BATTERY_LOAD/NUMBER_OF_BUSES)*ECLIPSE_TIME)
/(POWER_EQUINOX_CHARGE * CHARGE_DISCHARGE_EFFICIENCY_BATTERY);
PUT("Recharge time is ");
SET_COL(64);
PUT(RECHARGE_TIME,FORE=>3,AFT=>2,EXP=>0);
PUT(" hours");
NEW_LINE(2);

stop;

CREATE(OUTF,NAME=>"CELPARAM.DAT");

PUT_LINE(OUTF,"******************************************************************************");
SET_COL(OUTF,20);
PUT_LINE(OUTF,"ELECTRICAL POWER SUBSYSTEM PARAMETERS");
PUT_LINE(OUTF,"******************************************************************************");
NEW_LINE(OUTF,2);


SET_COL(OUTF,15);
PUT(OUTF,"PARAMETER");
SET_COL(OUTF,46);
PUT(OUTF,"SYMBOL");
SET_COL(OUTF,61);
PUT(OUTF,"VALUE");
SET_COL(OUTF,68);
PUT(OUTF,"UNITS");
NEW_LINE(OUTF,2);
PUT_LINE(OUTF,"******************************************************************************");


PUT( OUTF,"Payload Power Requirements");
SET_COL(OUTF,58);
PUT( OUTF,PAYLOAD_POWER, FORE => 4, AFT => 2, EXP => 0);
SET_COL(OUTF,69);
PUT( OUTF,"watts");
NEW_LINE(OUTF,1);



PUT( OUTF,"Housekeeping Power ");
SET_COL(OUTF,46);
PUT(OUTF,"Phk");
SET_COL(OUTF,58);
PUT( OUTF,HOUSEKEEPING_POWER, FORE => 4, AFT => 2, EXP => 0);
SET_COL(OUTF,69);
PUT( OUTF,"watts");
NEW_LINE(OUTF,1);
```

```
PUT( OUTF,"Load including load contingency");
SET_COL(OUTF,46);
PUT(OUTF,"Ptot");
SET_COL(OUTF,58);
PUT(OUTF,BATTERY_LOAD, FORE => 4, AFT => 2, EXP => 0);
SET_COL(OUTF,69);
PUT(OUTF,"watts");
NEW_LINE(OUTF,1);

PUT(OUTF,"Solar Array Load");
SET_COL(OUTF,46);
PUT(OUTF,"Psal");
SET_COL(OUTF,58);
PUT(OUTF,SOLAR_ARRAY_LOAD, FORE => 4, AFT => 2, EXP -> 0);
SET_COL(OUTF,69);
PUT(OUTF,"watts");
NEW_LINE(OUTF,1);

PUT(OUTF,"Spacecraft Life");
SET_COL(OUTF,58);
PUT(OUTF,SPACECRAFT_LIFE,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTF,69);
PUT(OUTF,"years");
NEW_LINE(OUTF,1);

PUT(OUTF,"Depth of Discharge");
SET_COL(OUTF,46);
PUT(OUTF,"DOD");
SET_COL(OUTF,58);
PUT(OUTF,DEPTH_OF_DISCHARGE,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTF,69);
PUT(OUTF,"%");
NEW_LINE(OUTF,1);

PUT(OUTF,"EOL Battery Discharge Voltage");
SET_COL(OUTF,46);
PUT(OUTF,"Vd");
set_col(OUTF,58);
PUT(OUTF,EOL_BATTERY_DISCHARGE_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTF,69);
PUT(OUTF,"volts");
NEW_LINE(OUTF,1);

PUT(OUTF,"Bypass Diode Voltage Drop");
SET_COL(OUTF,46);
PUT(OUTF,"Vdd");
SET_COL(OUTF,58);
PUT(OUTF,BYPASS_DIODE_VOLTAGE_DROP,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTF,69);
PUT(OUTF,"volts");
NEW_LINE(OUTF,1);
```

```
  PUT(OUTF,"Design Satellite Bus Voltage");
  SET_COL(OUTF,46);
  PUT(OUTF,"Vbus");
  SET_COL(OUTF,58);
  PUT(OUTF,BUS_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);
  SET_COL(OUTF,69);
  PUT(OUTF,"volts");
  NEW_LINE(OUTF,1);

  PUT(OUTF,"Bus Voltage Allowable Deviation");
  SET_COL(OUTF,46);
  PUT(OUTF,"Vbdev");
  SET_COL(OUTF,58);
  PUT(OUTF,BUS_VOLTAGE_ALLOWABLE_DEVIATION,FORE=>4,AFT=>2,EXP=>0);
  SET_COL(OUTF,69);
  PUT(OUTF,"volts");
  NEW_LINE(OUTF,1);

  PUT(OUTF,"Minimum Bus Voltage in Sunlight");
  SET_COL(OUTF,46);
  PUT(OUTF,"Vbmin");
  SET_COL(OUTF,58);

PUT(OUTF,BUS_VOLTAGE-BUS_VOLTAGE_ALLOWABLE_DEVIATION,FORE=>4,AFT=>2,EXP=>0);
  SET_COL(OUTF,69);
  PUT(OUTF,"volts");
  NEW_LINE(OUTF,1);



  PUT(OUTF,"Eclipse Time (Geosynchronous Orbits");
  SET_COL(OUTF,46);
  PUT(OUTF,"t");
  SET_COL(OUTF,58);
  PUT(OUTF,ECLIPSE_TIME,FORE=>4,AFT=>2,EXP=>0);
  SET_COL(OUTF,69);
  PUT(OUTF,"hours");
  NEW_LINE(OUTF,1);

  PUT(OUTF,"Maximum Battery Charge Voltage");
  SET_COL(OUTF,46);
  PUT(OUTF,"Vbc");
  SET_COL(OUTF,58);
  PUT(OUTF,MAXIMUM_CHARGE_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);
  SET_COL(OUTF,69);
  PUT(OUTF,"volts");
  NEW_LINE(OUTF,1);

  PUT(OUTF,"Number of Satellite Buses");
  SET_COL(OUTF,46);
  PUT(OUTF,"Buses");
```

```
SET_COL(OUTF,58);
PUT(OUTF,J,WIDTH=>4);
NEW_LINE(OUTF,1);



PUT(OUTF,"Bus Power");
SET_COL(OUTF,46);
PUT(OUTF,"Vbus");
SET_COL(OUTF,58);
PUT(OUTF,BUS_POWER,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTF,69);
PUT(OUTF,"watts");
NEW_LINE(OUTF,1);


PUT(OUTF,"Number of Series Connected Diodes");
SET_COL(OUTF,46);
PUT(OUTF,"Ndiode");
SET_COL(OUTF,58);
PUT(OUTF,NUMBER_SERIES_CONNECTED_DIODES,FORE=>4,AFT=>2,EXP=>0);
NEW_LINE(OUTF,1);

PUT(OUTF,"Series Connected Diode Voltage Drop");
SET_COL(OUTF,46);
PUT(OUTF,"Vdiode");
SET_COL(OUTF,58);
PUT(OUTF,SERIES_CONNECTED_DIODE_VOLTAGE_DROP,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTF,69);
PUT(OUTF,"volts");
NEW_LINE(OUTF,1);

PUT(OUTF,"Battery Charger Voltage Drop");
SET_COL(OUTF,46);
PUT(OUTF,"Vcd");
SET_COL(OUTF,58);
PUT(OUTF,BATTERY_CHARGER_VOLTAGE_DROP,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTF,69);
PUT(OUTF,"volts");
NEW_LINE(OUTF,1);

PUT(OUTF,"Equinox Current");
SET_COL(OUTF,46);
PUT(OUTF,"Ieq");
SET_COL(OUTF,58);
PUT(OUTF,EQUINOX_CURRENT,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTF,69);
PUT(OUTF,"amps");
NEW_LINE(OUTF,1);

PUT(OUTF,"Solstice Current");
SET_COL(OUTF,46);
```

```
PUT(OUTF,"Iss");
SET_COL(OUTF,58);
PUT(OUTF,SOLSTICE_CURRENT,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTF,69);
PUT(OUTF,"amps");
NEW_LINE(OUTF,1);

PUT(OUTF,"Battery Charge Discharge Efficiency");
SET_COL(OUTF,46);
SET_COL(OUTF,58);
PUT(OUTF,CHARGE_DISCHARGE_EFFICIENCY_BATTERY,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTF,69);
PUT(OUTF,"%");

NEW_LINE(OUTF,1);

PUT(OUTF,"Number of Satellite Buses");
SET_COL(OUTF,58);
PUT(OUTF,J,WIDTH=>4);
NEW_LINE(OUTF,1);

PUT(OUTF,"Number of Battery Cells in Series");
SET_COL(OUTF,46);
PUT(OUTF,"N");
SET_COL(OUTF,58);
PUT(OUTF,SERIES_CELLS_FOR_MIN_DIS_VOLT,FORE=>4,AFT=>2,EXP=>0);
NEW_LINE(OUTF,1);

PUT(OUTF,"Minimum Discharge Bus Voltage");
SET_COL(OUTF,46);
PUT(OUTF,"Vdb");
SET_COL(OUTF,58);
PUT(OUTF,MINIMUM_DISCHARGE_BUS_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTF,69);
PUT(OUTF,"volts");
NEW_LINE(OUTF,1);

PUT(OUTF,"Required Cell Capacity");
SET_COL(OUTF,46);
PUT(OUTF,"C");
SET_COL(OUTF,58);
PUT(OUTF,CELL_AH,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTF,69);
PUT(OUTF,"AH");
NEW_LINE(OUTF,1);


PUT(OUTF,"Boost Voltage Req. by Charge Array");
SET_COL(OUTF,46);
PUT(OUTF,"Vca");
SET_COL(OUTF,58);
PUT(OUTF,VOLTAGE_CHARGE_ARRAY, FORE=>4,AFT=>2,EXP=>0);
```

```
  SET_COL(OUTF,69);
  PUT(OUTF,"volts");
  NEW_LINE(OUTF,1);

  PUT(OUTF,"Maximum Battery Charge Voltage");
  SET_COL(OUTF,46);
  PUT(OUTF,"Vbc");
  SET_COL(OUTF,58);

PUT(OUTF,BUS_VOLTAGE+MAXIMUM_BATTERY_CHARGE_VOLTAGE,FORE=>4,AFT=>2,EXP=
>0);
  SET_COL(OUTF,69);
  PUT(OUTF,"volts");
  NEW_LINE(OUTF,1);

  PUT(OUTF,"Power Req. Equinox Charge @ High Rate");
  SET_COL(OUTF,46);
  PUT(OUTF,"Pchrgeq");
  SET_COL(OUTF,58);
  PUT(OUTF,POWER_EQUINOX_CHARGE,FORE=>4,AFT=>2,EXP=>0);
  SET_COL(OUTF,69);
  PUT(OUTF,"watts");
  NEW_LINE(OUTF,1);

  PUT(OUTF,"Power Req. Solstice Charge @ High Rate");
  SET_COL(OUTF,46);
  PUT(OUTF,"Pchrgss");
  SET_COL(OUTF,58);
  PUT(OUTF,POWER_SOLSTICE_CHARGE,FORE=>4,AFT=>2,EXP=>0);
  SET_COL(OUTF,69);
  PUT(OUTF,"watts");
  NEW_LINE(OUTF,1);

  PUT(OUTF,"Recharge Time");
  SET_COL(OUTF,46);
  PUT(OUTF,"Tcharge-e");
  SET_COL(OUTF,58);
  PUT(OUTF,RECHARGE_TIME,FORE=>4,AFT=>2,EXP=>0);
  SET_COL(OUTF,69);
  PUT(OUTF,"hours");
  NEW_LINE(OUTF,1);
  PUT_LINE(OUTF,"******************************************************************************");
  PUT_LINE(OUTF,"******************************************************************************");


  CLOSE(OUTF);
```

```
-- MATERIAL,


 VIDEO.CLEAR_SCREEN;
 PUT_LINE("SOLAR ARRAY DESIGN");
 PUT_LINE("  The total load on the solar array will be the summation of");
 PUT_LINE("the equipment  load and the power  required for  charging the");
 PUT_LINE("batteries.  Generally, 10% solar  array design margin is used");
 PUT_LINE("to  take  into  account  the  uncertainty  in  the  radiation ");
 PUT_LINE("degradation and other design factors of the solar array.");

 -- should I use payload power or bus power



 DESIGN_LOAD_EQUINOX:=DESIGN_MARGIN*(BATTERY_LOAD+POWER_EQUINOX_CHARGE);
 NEW_LINE(2);
 PUT("The solar array design load at EQUINOX is  ");
 SET_COL(64);
 PUT(DESIGN_LOAD_EQUINOX,FORE=>5,AFT=>2,EXP=>0);PUT("  watts");

 DESIGN_LOAD_SOLSTICE:=DESIGN_MARGIN*(BATTERY_LOAD+POWER_SOLSTICE_CHARGE);
 NEW_LINE(2);
 PUT("The solar array design load at SOLSTICE is  ");
 SET_COL(64);
 PUT(DESIGN_LOAD_SOLSTICE,FORE=>5,AFT=>2,EXP=>0);PUT("  watts");
 NEW_LINE(2);
 PUT_LINE("  The design load for either configuration will be divided ");
 PUT_LINE("by the total number of buses for the main load.");
 NEW_LINE(2);

 STOP;
 NEW_LINE(1);

CELL_CURRENT_EOL_SOLSTICE:=((CURRENT_MAX_POWER
                +TEMP_COEF_EOL_CURRENT
                *(SOLSTICE_SOLAR_ARRAY_TEMP-SOLAR_CELL_TEST_TEMP))
                *ASSEMBLY_LOSS_CURRENT* ENVIRONMENTAL_DEGRADATION_CURRENT
                *SOLSTICE_SIF_CURRENT);
PUT_LINE("Solar Cell Current at Maximum Power Point ");
PUT("EOL Summer Solstice I-V curve is  ");
SET_COL(60);
PUT(CELL_CURRENT_EOL_SOLSTICE, FORE=>3,AFT=>4,EXP=>0);PUT("  Amps");
NEW_LINE(2);

CELL_CURRENT_EOL_EQUINOX:=((CURRENT_MAX_POWER
                +TEMP_COEF_EOL_CURRENT
                *(EQUINOX_SOLAR_ARRAY_TEMP-SOLAR_CELL_TEST_TEMP))
                *ASSEMBLY_LOSS_CURRENT* ENVIRONMENTAL_DEGRADATION_CURRENT
                *EQUINOX_SIF_CURRENT);
PUT_LINE("Solar Cell Current at Maximum Power Point ");
PUT("EOL Autumnal Equinox I-V curve is  ");
```

```
 SET_COL(60);
 PUT(CELL_CURRENT_EOL_EQUINOX, FORE=>3,AFT=>4,EXP=>0);PUT("  Amps");
 NEW_LINE(2);

 REQUIRED_CURRENT_SOLSTICE_PER_BUS:=DESIGN_LOAD_SOLSTICE/NUMBER_OF_BUSES
                                              /BUS_VOLTAGE;
 PUT("Required Current Solstice Per Bus");
 SET_COL(60);
 PUT(REQUIRED_CURRENT_SOLSTICE_PER_BUS,FORE=>3,AFT=>4,EXP=>0);
 PUT("  amps");
 NEW_LINE(2);

 REQUIRED_CURRENT_EQUINOX_PER_BUS:=DESIGN_LOAD_EQUINOX/NUMBER_OF_BUSES
                                              /BUS_VOLTAGE;
 PUT_LINE("Required Current Equinox Per Bus");
 SET_COL(60);
 PUT(REQUIRED_CURRENT_EQUINOX_PER_BUS,FORE=>3,AFT=>4,EXP=>0);
 PUT("  amps");
 NEW_LINE(2);

 PUT("BUS VOLTAGE");
 SET_COL(60);
 PUT(BUS_VOLTAGE,FORE=>3,AFT=>4,EXP=>0);
 NEW_LINE(4);

 STOP;

 NEW_LINE(1);
 NUMBER_CELLS_IN_PARALLEL:=
              REQUIRED_CURRENT_SOLSTICE_PER_BUS/CELL_CURRENT_EOL_SOLSTICE;
 PUT("# cells in parallel for minimum current each bus is");
 SET_COL(65);
 PUT(NUMBER_CELLS_IN_PARALLEL,FORE=>4,AFT=>2,EXP=>0);
 NEW_LINE(1);
N_INTEGER:=INTEGER(NUMBER_CELLS_IN_PARALLEL);
PUT("# cells in parallel for minimum current INTEGER");
SET_COL(65);
PUT(FLOAT(N_INTEGER),FORE=>4.AFT=>2,EXP=>0);
NEW_LINE(1);

if FLOAT(N_INTEGER) < NUMBER_CELLS_IN_PARALLEL then
  N_INTEGER:=N_INTEGER+1;
end if;

NUMBER_CELLS_IN_PARALLEL:=FLOAT(N_INTEGER);
PUT("# cells in parallel for minimum current ROUND UP ");
SET_COL(65);
PUT(NUMBER_CELLS_IN_PARALLEL,FORE=>4,AFT=>2,EXP=>0);
new_line(2);

--TEMP_OF_CELL_CHARACS
  CELL_VOLTAGE_EOL_SOLSTICE:=(VOLTAGE_MAX_POWER-PANEL_WIRING_LOSS_PER_CELL
                +TEMP_COEF_EOL_VOLTAGE
                *(SOLSTICE_SOLAR_ARRAY_TEMP-SOLAR_CELL_TEST_TEMP))
                *ENVIRONMENTAL_DEGRADATION_VOLTAGE;
 PUT("Solar Cell Voltage at EOL Solstice is ");
 SET_COL(65);
 PUT(CELL_VOLTAGE_EOL_SOLSTICE,FORE=>1,AFT=>4,EXP=>0);PUT("  volts");
 NEW_LINE(2);

 CELL_VOLTAGE_EOL_EQUINOX:=(VOLTAGE_MAX_POWER-PANEL_WIRING_LOSS_PER_CELL
                +TEMP_COEF_EOL_VOLTAGE
                *(EQUINOX_SOLAR_ARRAY_TEMP-SOLAR_CELL_TEST_TEMP))
                *ENVIRONMENTAL_DEGRADATION_VOLTAGE;
 PUT("Solar Cell Voltage at EOL Equinox is ");
 SET_COL(65);
 PUT(CELL_VOLTAGE_EOL_EQUINOX,FORE=>1,AFT=>4,EXP=>0);PUT("  volts");
 NEW_LINE(2);


 NUMBER_CELLS_IN_SERIES:=(BUS_VOLTAGE + BLOCKING_DIODE_VOLTAGE_DROP
                        + ARRAY_WIRING_HARNESS_AND_SLIP_RING_VOLTAGE_DROP)
                        /CELL_VOLTAGE_EOL_SOLSTICE;
 PUT("# series cells for minimum discharge voltage");
 SET_COL(65);
   PUT(NUMBER_CELLS_IN_SERIES,FORE=>4,AFT=>2,EXP=>0);
   new_line(1);
   N_INTEGER:=INTEGER(NUMBER_CELLS_IN_SERIES);
   PUT("# series cells for minimum discharge voltage INTEGER");
   SET_COL(65);
   PUT(FLOAT(N_INTEGER),FORE=>4,AFT=>2,EXP=>0);
   NEW_LINE(1);
   if FLOAT(N_INTEGER) < NUMBER_CELLS_IN_SERIES  then
       N_INTEGER:=N_INTEGER+1;
   end if;
   NUMBER_CELLS_IN_SERIES:=FLOAT(N_INTEGER);
   -- minimum discharge voltage

   PUT("# series cells for minimum discharge voltage ROUND UP");
   SET_COL(65);
   PUT(NUMBER_CELLS_IN_SERIES,FORE=>4,AFT=>2,EXP=>0);
   new_line(4);

   STOP;

-- Cell current at maximum power point of EOL autumnal equinox

NEW_LINE(1);
CELL_CURRENT_EOL_EQUINOX:= (CURRENT_MAX_POWER+TEMP_COEF_EOL_CURRENT
                        *(EQUINOX_SOLAR_ARRAY_TEMP-SOLAR_CELL_TEST_TEMP))
                        *ASSEMBLY_LOSS_CURRENT* ENVIRONMENTAL_DEGRADATION_CURRENT
                        *EQUINOX_SIF_CURRENT;
PUT("Solar Cell Current at EOL Autumnal Equinox  ");
SET_COL(60);
```

```
PUT(CELL_CURRENT_EOL_EQUINOX, FORE=>1,AFT=>4,EXP=>0);PUT(" Amps");
NEW_LINE(2);

 CELL_VOLTAGE_EOL_EQUINOX:=(VOLTAGE_MAX_POWER-PANEL_WIRING_LOSS_PER_CELL
                   +TEMP_COEF_EOL_VOLTAGE
                   *(EQUINOX_SOLAR_ARRAY_TEMP-SOLAR_CELL_TEST_TEMP))
                   *ENVIRONMENTAL_DEGRADATION_VOLTAGE;
  PUT("Solar Cell Voltage at EOL equinox is ");
  SET_COL(60);
  PUT(CELL_VOLTAGE_EOL_EQUINOX,FORE=>1,AFT=>4,EXP=>0);PUT(" volts");
  NEW_LINE(2);


 BUS_CURRENT:=CELL_CURRENT_EOL_EQUINOX*NUMBER_CELLS_IN_PARALLEL;
 PUT("The current per bus or wing is ");
 SET_COL(60);
 PUT(BUS_CURRENT, FORE=>3,AFT=>2,EXP=>0);
 PUT(" amps");
 NEW_LINE(2);

 BUS_VOLTAGE:=(CELL_VOLTAGE_EOL_EQUINOX*NUMBER_CELLS_IN_SERIES)
            -(BLOCKING_DIODE_VOLTAGE_DROP
             +ARRAY_WIRING_HARNESS_AND_SLIP_RING_VOLTAGE_DROP);
 PUT("The voltage per bus or wing is ");
 SET_COL(60);
 PUT(BUS_VOLTAGE, FORE=>3,AFT=>2,EXP=>0);
 PUT(" volts");
 NEW_LINE(2);

 TOTAL_POWER_PER_BUS:=BUS_CURRENT*BUS_VOLTAGE;
 ARRAY_TOTAL_POWER:=TOTAL_POWER_PER_BUS*NUMBER_OF_BUSES;
 PUT("The Total Power is ");
 SET_COL(58);
 PUT(ARRAY_TOTAL_POWER, FORE=>5,AFT=>2,EXP=>0);
 PUT(" Watts");
 NEW_LINE(4);
 STOP;


--**********************************************************************

  if DRUM_SPINNER = FALSE then
    POWER_MARGIN:=DESIGN_MARGIN*POWER_TOTAL;
  else
    POWER_MARGIN:=DESIGN_MARGIN*POWER_TOTAL/PI;
  end if;

  if ARRAY_TOTAL_POWER >= DESIGN_LOAD_EQUINOX and DRUM_SPINNER = TRUE then
    if (ARRAY_TOTAL_POWER-DESIGN_LOAD_EQUINOX) > 100.0 then
        VIDEO.CLEAR_SCREEN;
        PUT("Total array power is ");
        PUT(ARRAY_TOTAL_POWER-DESIGN_LOAD_EQUINOX,FORE=>4,AFT=>2,EXP=>0);
        PUT("  watts more than needed.");
        NEW_LINE(1);
        PUT_LINE("If substantially greater recommend DESIGN BUS VOLTAGE be reduced");
        PUT_LINE("Recommend re-iterate to optimize design ");
        NEW_LINE(1);
        FINAL:=FALSE;
        GET_CHARACTER(CHAR);
        if CHAR = 'Y' or CHAR = 'y' then
          CHAR:=N;
          VIDEO.CLEAR_SCREEN;
          goto  REITERATE_VOLTAGE;
        else
         goto MOVEON;  --  <<MOVEON>> Exit if structure
        end if;
    end if;

  elsif  ARRAY_TOTAL_POWER < DESIGN_LOAD_EQUINOX then
          PUT("THE TOTAL ARRAY POWER IS ");

PUT(ABS(DESIGN_LOAD_EQUINOX-ARRAY_TOTAL_POWER),FORE=>4,AFT=>2,EXP=>0);
          PUT("  watts less than needed.");
          NEW_LINE(2);
          PUT("Recommend increasing DESIGN BUS VOLTAGE");
          new_line(2);
          PUT_LINE("Recommend re-iterate to optimize design ");
          FINAL:=FALSE;
          SET_COL(10);
          GET_CHARACTER(CHAR);
          if CHAR = 'Y' or CHAR = 'y' then
            CHAR:=N;
            VIDEO.CLEAR_SCREEN;
            goto  REITERATE_VOLTAGE;
          else
          goto MOVEON;  --  <<MOVEON>> Exit if structure
          end if;

  elsif  (ARRAY_TOTAL_POWER-DESIGN_LOAD_EQUINOX) < 100.0
         and DRUM_SPINNER = FALSE
         and (ARRAY_TOTAL_POWER-DESIGN_LOAD_EQUINOX) > 0.0 then

     VIDEO.CLEAR_SCREEN;

     PUT_LINE("Design Bus Voltage and Minimum Discharge Bus Voltage");
     PUT_LINE("are within optimal design parameters.");
     NEW_LINE(4);
     STOP;
  end if;

 <<MOVEON>>
```

```
--            CHARGE ARRAY DESIGN
NEW_LINE(1);
STOP;
PUT_LINE("                    CHARGE ARRAY DESIGN    ");
PUT_LINE("-----------------------------------------------------------------");

NUMBER_CELLS_IN_SERIES_CHARGE_ARRAY_SOLSTICE:=
                                    VOLTAGE_CHARGE_ARRAY/CELL_VOLTAGE_EOL_SOLSTICE;
NEW_LINE(1);
PUT("# of series cells for charge array during solstice, Nc, is ");
SET_COL(70);
PUT(NUMBER_CELLS_IN_SERIES_CHARGE_ARRAY_SOLSTICE,FORE=>3,AFT=>2,EXP=>0);
NEW_LINE(1);
N_INTEGER:=INTEGER(NUMBER_CELLS_IN_SERIES_CHARGE_ARRAY_SOLSTICE);
PUT("# series cells charge  during solstice, Nc, INTEGER is ");
SET_COL(70);
PUT(FLOAT(N_INTEGER),FORE=>3,AFT=>2,EXP=>0);
NEW_LINE(1);
if FLOAT(N_INTEGER) < NUMBER_CELLS_IN_SERIES_CHARGE_ARRAY_SOLSTICE  then
  N_INTEGER:=N_INTEGER+1;
end if;
NUMBER_CELLS_IN_SERIES_CHARGE_ARRAY_SOLSTICE:=FLOAT(N_INTEGER);
SET_COL(1);
PUT("# series cells charge  during solstice, Nc, ROUND UP is  ");
SET_COL(70);
PUT(NUMBER_CELLS_IN_SERIES_CHARGE_ARRAY_SOLSTICE,FORE=>3,AFT=>2,EXP=>0);
new_line(2);

-- Parallel  ells charge array solstice


NUMBER_CELLS_IN_PARALLEL_CHARGE_ARRAY_SOLSTICE:=
                    CELL_AH/SOLSTICE_CHARGE_RATE/CELL_CURRENT_EOL_SOLSTICE;

SET_COL(1);
PUT("# parallel cells charge array during solstice, Ncs, is ");
SET_COL(70);
PUT(NUMBER_CELLS_IN_PARALLEL_CHARGE_ARRAY_SOLSTICE,FORE=>3,AFT=>2,EXP=>0);
NEW_LINE(1);
N_INTEGER:=INTEGER(NUMBER_CELLS_IN_PARALLEL_CHARGE_ARRAY_SOLSTICE);
PUT("# parallel cells charge array during solstice, Ncs, INTEGER is ");
SET_COL(70);
PUT(FLOAT(N_INTEGER),FORE=>3,AFT=>2,EXP=>0);

NEW_LINE(1);
if FLOAT(N_INTEGER) < NUMBER_CELLS_IN_PARALLEL_CHARGE_ARRAY_SOLSTICE then
  N_INTEGER:=N_INTEGER+1;
end if;
```

```
NUMBER_CELLS_IN_PARALLEL_CHARGE_ARRAY_SOLSTICE:=FLOAT(N_INTEGER);
PUT("# parallel cells charging array during solstice, Ncs, ROUND UP is");
SET_COL(70);
PUT(NUMBER_CELLS_IN_PARALLEL_CHARGE_ARRAY_SOLSTICE,FORE=>3,AFT=>2,EXP=>0);
NEW_LINE(2);

-- Parallel cells charge array equinox

NUMBER_CELLS_IN_PARALLEL_CHARGE_ARRAY_EQUINOX:=
              CELL_AH/EQUINOX_CHARGE_RATE/CELL_CURRENT_EOL_EQUINOX;


PUT("# parallel cells charge array during equinox, Nce, is  ");
SET_COL(70);
PUT(NUMBER_CELLS_IN_PARALLEL_CHARGE_ARRAY_EQUINOX,FORE=>3,AFT=>2,EXP=>0);
NEW_LINE(1);
N_INTEGER:=INTEGER(NUMBER_CELLS_IN_PARALLEL_CHARGE_ARRAY_EQUINOX);

PUT("# parallel cells charge array during equinox, Nce, INTEGER is  ");
SET_COL(70);
PUT(FLOAT(N_INTEGER),FORE=>3,AFT=>2,EXP=>0);
NEW_LINE(1);
if FLOAT(N_INTEGER) < NUMBER_CELLS_IN_PARALLEL_CHARGE_ARRAY_EQUINOX   then
  N_INTEGER:=N_INTEGER+1;
end if;

NUMBER_CELLS_IN_PARALLEL_CHARGE_ARRAY_EQUINOX:=FLOAT(N_INTEGER);
PUT("# parallel cells charge array during equinox, Nce, ROUND UP is  ");
SET_COL(70);
PUT(NUMBER_CELLS_IN_PARALLEL_CHARGE_ARRAY_EQUINOX,FORE=>3,AFT=>2,EXP=>0);
NEW_LINE(4);

--*************************************************************************

  CREATE(OUTSC,NAME=>"SOLCELL.DAT");

  SET_LINE(OUTSC,1);
  PUT_LINE(OUTSC,"**************************************************************************");
  SET_COL(OUTSC,20);
  PUT(OUTSC,"SOLAR CELL CHARACTERISTICS");
  new_line(OUTSC,1);
  PUT_LINE(OUTSC,"**************************************************************************");
  NEW_LINE(OUTSC,1);

  SET_COL(OUTSC,15);
  PUT(OUTSC,"CHARACTERISTIC");
  SET_COL(OUTSC,47);
  PUT(OUTSC,"VALUE");
  SET_COL(OUTSC,59);
  PUT(OUTSC,"UNITS");
  NEW_LINE(OUTSC,1);
  PUT_LINE(OUTSC,"**************************************************************************");
```

```
PUT(OUTSC,"TEMPERATURE BASIS FOR PERFORMANCE");
SET_COL(OUTSC,44);
PUT(OUTSC,SOLAR_CELL_TEST_TEMP ,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"deg celcius");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"TEMPERATURE SOLAR ARRAY SOLSTICE");


SET_COL(OUTSC,44);
PUT(OUTSC,SOLAR_ARRAY_TEMP_SOLSTICE,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"deg celcius ");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"SOLAR ARRAY TEMPERATURE EQUINOX");


SET_COL(OUTSC,44);
PUT(OUTSC,SOLAR_ARRAY_TEMP_EQUINOX,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"deg celcius ");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"TEMPERATURE COEFFICIENT EOL CURRENT ");
SET_COL(OUTSC,44);
PUT(OUTSC,TEMP_COEF_EOL_CURRENT,FORE=>1,AFT=>5,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"ma/deg celcius");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"TEMPERATURE COEFFICIENT EOL VOLTAGE ");
SET_COL(OUTSC,43);
PUT(OUTSC,TEMP_COEF_EOL_VOLTAGE,FORE=>2,AFT=>5,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"mv/deg celcius");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"POWER PER SOLAR CELL");


SET_COL(OUTSC,44);
PUT(OUTSC,CELL_CURRENT_EOL_EQUINOX*CELL_VOLTAGE_EOL_EQUINOX
                                        ,FORE=>3,AFT=>3,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"watts");
NEW_LINE(OUTSC,1);
```

```
PUT(OUTSC,"CELL CURRENT @ MAX POWER POINT");
SET_COL(OUTSC,44);
PUT(OUTSC,CELL_CURRENT_EOL_EQUINOX ,FORE=>3,AFT=>3,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"amps");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"CELL VOLTAGE @ MAX POWER POINT ");
SET_COL(OUTSC,44);
PUT(OUTSC,CELL_VOLTAGE_EOL_EQUINOX ,FORE=>3,AFT=>3,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"volts");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"CURRENT - SHORT CIRCUIT");
SET_COL(OUTSC,44);
PUT(OUTSC,CURRENT_SHORT_CIRCUIT ,FORE=>3,AFT=>3,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"amps");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"VOLTAGE - OPEN CIRCUIT");
SET_COL(OUTSC,44);
PUT(OUTSC,VOLTAGE_OPEN_CIRCUIT,FORE=>3,AFT=>3,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"volts");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"CELL WIDTH");
SET_COL(OUTSC,44);
PUT(OUTSC,CELL_WIDTH,FORE=>3,AFT=>3,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"cm ");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"CELL LENGTH");
SET_COL(OUTSC,44);
PUT(OUTSC,CELL_LENGTH,FORE=>3,AFT=>3,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"cm ");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"CELL THICKNESS");
SET_COL(OUTSC,44);
PUT(OUTSC,CELL_THICKNESS,FORE=>3,AFT=>3,EXP=>0);
SET_COL(OUTSC,58);
PUT(OUTSC,"cm ");
NEW_LINE(OUTSC,1);

-- SOLAR ARRAY DATA

SET_LINE(OUTSC,1);
```

```
PUT_LINE(OUTSC,"****************************************************************************");
SET_COL(OUTSC,20);
PUT(OUTSC,"SOLAR ARRAY SYSTEM CHARACTERISTICS");
new_line(OUTSC,1);
PUT_LINE(OUTSC,"****************************************************************************");
NEW_LINE(OUTSC,1);

SET_COL(OUTSC,15);
PUT(OUTSC,"CHARACTERISTIC");
SET_COL(OUTSC,47);
PUT(OUTSC,"VALUE");
SET_COL(OUTSC,59);
PUT(OUTSC,"UNITS");
NEW_LINE(OUTSC,1);
PUT_LINE(OUTSC,"****************************************************************************");



PUT(OUTSC,"DESIGN MARGIN");
SET_COL(OUTSC,44);
PUT(OUTSC,DESIGN_MARGIN,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"%");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"DESIGN LOAD EQUINOX");
SET_COL(OUTSC,44);
PUT(OUTSC,DESIGN_LOAD_EQUINOX,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"watts");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"DESIGN LOAD SOLSTICE");
SET_COL(OUTSC,44);
PUT(OUTSC,DESIGN_LOAD_SOLSTICE,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"watts");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"CELL CURRENT EOL SOLSTICE");
SET_COL(OUTSC,44);
PUT(OUTSC,CELL_CURRENT_EOL_SOLSTICE,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"amps");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"REQUIRED CURRENT SOLSTICE PER BUS");
SET_COL(OUTSC,44);
PUT(OUTSC,REQUIRED_CURRENT_SOLSTICE_PER_BUS,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"amps");
```

```
NEW_LINE(OUTSC,1);

PUT(OUTSC,"REQUIRED CURRENT EQUINOX PER BUS");
SET_COL(OUTSC,44);
PUT(OUTSC,REQUIRED_CURRENT_EQUINOX_PER_BUS,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"amps");
NEW_LINE(OUTSC,1);


PUT(OUTSC,"NUMBER CELLS IN PARALLEL (FLAT PANEL)");
SET_COL(OUTSC,44);
PUT(OUTSC,NUMBER_CELLS_IN_PARALLEL,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"solar cells");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"NUMBER CELLS IN PARALLEL (CYLINDER)");
NUMBER_CELLS_IN_PARALLEL:=NUMBER_CELLS_IN_PARALLEL*PI;
N_INTEGER:=INTEGER(NUMBER_CELLS_IN_PARALLEL);

if FLOAT(N_INTEGER) < NUMBER_CELLS_IN_PARALLEL then
  N_INTEGER:=N_INTEGER+1;
end if;

NUMBER_CELLS_IN_PARALLEL:=FLOAT(N_INTEGER);
SET_COL(OUTSC,44);
PUT(OUTSC,NUMBER_CELLS_IN_PARALLEL,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"solar cells");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"CELL VOLTAGE EOL SOLSTICE");
SET_COL(OUTSC,44);
PUT(OUTSC,CELL_VOLTAGE_EOL_SOLSTICE,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"volts");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"NUMBER CELLS IN SERIES - PER BUS");
SET_COL(OUTSC,44);
PUT(OUTSC,NUMBER_CELLS_IN_SERIES,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"solar cells");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"TOTAL NUMBER OF CELLS");
SET_COL(OUTSC,44);
PUT(OUTSC,NUMBER_CELLS_IN_SERIES*NUMBER_CELLS_IN_PARALLEL
                                            ,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"solar cells");
```

```
NEW_LINE(OUTSC,1);

PUT(OUTSC,"CELL CURRENT EOL EQUINOX");
SET_COL(OUTSC,44);
PUT(OUTSC,CELL_CURRENT_EOL_EQUINOX,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"amps");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"REQUIRED BUS CURRENT EOL EQUINOX");
SET_COL(OUTSC,44);
PUT(OUTSC,BUS_CURRENT,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"amps");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"CELL VOLTAGE EOL EQUINOX");
SET_COL(OUTSC,44);
PUT(OUTSC,CELL_VOLTAGE_EOL_EQUINOX,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"volts");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"REQUIRED BUS VOLTAGE EOL EQUINOX");
SET_COL(OUTSC,44);
PUT(OUTSC,BUS_VOLTAGE,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"volts");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"TOTAL POWER PER BUS");
TOTAL_POWER_PER_BUS:=BUS_CURRENT*BUS_VOLTAGE;
SET_COL(OUTSC,44);
PUT(OUTSC,TOTAL_POWER_PER_BUS,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"watts");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"TOTAL POWER AVAILABLE FROM ARRAY");
POWER_TOTAL:=TOTAL_POWER_PER_BUS*NUMBER_OF_BUSES;
SET_COL(OUTSC,44);
PUT(OUTSC,POWER_TOTAL,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"watts");
NEW_LINE(OUTSC,1);

PUT(OUTSC,"POWER MARGIN 3-AXIS STABILIZED");
POWER_MARGIN:=POWER_TOTAL*(DESIGN_MARGIN-1.0);
SET_COL(OUTSC,44);
PUT(OUTSC,POWER_MARGIN,FORE=>4,AFT=>2,EXP=>0);
SET_COL(OUTSC,59);
PUT(OUTSC,"watts");
```

```
  NEW_LINE(OUTSC,1);



  PUT(OUTSC,"POWER MARGIN DUAL-SPIN");
  POWER_MARGIN:=POWER_TOTAL*(DESIGN_MARGIN-1.0)/PI;
  SET_COL(OUTSC,44);
  PUT(OUTSC,POWER_MARGIN,FORE=>4,AFT=>2,EXP=>0);
  SET_COL(OUTSC,59);
  PUT(OUTSC,"watts");
  NEW_LINE(OUTSC,1);



  CLOSE (OUTSC);

end battery;

begin -- MAIN

PRINT_HEADER;

  DUAL_SPIN           (DRUM_SPINNER);

  OPERATING_DATA         (BATTERY_LOAD,
                       DRUM_SPINNER,
                       MINIMUM_DISCHARGE_BUS_VOLTAGE,
                       BUS_VOLTAGE,
                       BYPASS_DIODE_VOLTAGE_DROP,
                       EOL_BATTERY_DISCHARGE_VOLTAGE,
                       BUS_POWER,
                       PAYLOAD_POWER,
                       DEPTH_OF_DISCHARGE,
                       number_of_buses,
                       SPACECRAFT_LIFE,
                       ECLIPSE_TIME);

  BATTERY             (BATTERY_LOAD,
                       DRUM_SPINNER,
                       MINIMUM_DISCHARGE_BUS_VOLTAGE,
                       BUS_VOLTAGE,
                       BYPASS_DIODE_VOLTAGE_DROP,
                       EOL_BATTERY_DISCHARGE_VOLTAGE,
                       BUS_POWER,
                       PAYLOAD_POWER,
                       CELL_AH,
                       DEPTH_OF_DISCHARGE,
                       ECLIPSE_TIME,
                       VOLTAGE_CHARGE_ARRAY,
                       NUMBER_OF_BUSES,
                                MAXIMUM_BATTERY_CHARGE_VOLTAGE,
                                SERIES_CONNECTED_DIODE_VOLTAGE_DROP,
                                NUMBER_SERIES_CONNECTED_DIODES,
                                BATTERY_CHARGER_VOLTAGE_DROP,
                                CHARGE_DISCHARGE_EFFICIENCY_BATTERY,
                                spacecraft_life,
                                POWER_EQUINOX_CHARGE,
                                POWER_SOLSTICE_CHARGE);



  SET_COL(10);PUT("TO CONTINUE ENTER ANY INTEGER"); GET_INTEGER(I);
  VIDEO.CLEAR_SCREEN;
  NEW_LINE(2);
  PUT_LINE("DATA FILES FOR THIS DESIGN RUN ARE LOCATED IN THE FOLLOWING FILES:");
  NEW_LINE(1);
  PUT_LINE("                    CELPARAM.DAT");
  PUT_LINE("                    SOLCELL.DAT ");
  NEW_LINE(1);
  PUT_LINE("TO KEEP DATA FROM BEING ERASED ON NEXT RUN");
  PUT_LINE("USE DOS COMMAND REN (RENAME) ");
  NEW_LINE(1);
  PUT_LINE("EXAMPLE - REN SOLARCEL.DAT   SOLARCEL.XYZ");

end SOLARPOWER;
```

C. PASSIVE THERMAL CONTROL

```
-- Title        : Thermal Characteristics
-- Author        : David Lashbrook
-- Date         : 15 February 1992
-- Revised       : 12 May 1992
-- Compiler       : OPENADA EXT
-- Description   : This procedure determines the delta velocity for insertion
--              into geosynchronous orbit.
with TEXT_IO, GENERIC_ELEMENTARY_FUNCTIONS, GETDATA,VIDEO;
use  TEXT_IO, GETDATA ;
procedure THERMAL is     -- THERMAL_CONTROL

  package FLOAT_INOUT is new FLOAT_IO(FLOAT);
  use    FLOAT_INOUT;
  package INTEGER_INOUT is new INTEGER_IO(INTEGER);
  use    INTEGER_INOUT;
  package BOOLEAN_INOUT is new ENUMERATION_IO(BOOLEAN);
  use    BOOLEAN_INOUT;
  package GEF_INOUT is new GENERIC_ELEMENTARY_FUNCTIONS(FLOAT);
  use    GEF_INOUT;

  DEPTH_OF_DISCHARGE            : FLOAT:=0.65;
  ECLIPSE_TIME               : FLOAT:=1.20; -- hours
  TIME_ECLIPSE               : FLOAT:=72.0; -- minutes
  PI                      : constant FLOAT := 3.14159265359;
  SPACECRAFT_LIFE,
  SPACECRAFT_MASS_BEFORE_APOGEE_BURN,
  PAYLOAD_POWER,
  BATTERY_LOAD,
  NUMBER_OF_BUSES,
  SOLAR_ARRAY_LOAD            : FLOAT;

  LIFE_FACTOR             : FLOAT   := 1.05;      --
  POWER_MARGIN              : FLOAT   := 1.1;       -- margin for error
  RADIATOR_SPECIFIC_HEAT       : FLOAT   := 900.0; -- (watts*sec)\(kg*Kelvin)
  INTELSAT_7_REFERENCE        : FLOAT   :=3445.0;     -- kgs
  INTELSAT_6_REFERENCE        : FLOAT   :=2227.0;     -- kgs
  INTELSAT_5_REFERENCE        : FLOAT   :=1900.0;     --
INTELSAT_7_HOUSEKEEPING_POWER: constant FLOAT := 613.0; -- intelsat VII
INTELSAT_6_HOUSEKEEPING_POWER: constant FLOAT := 347.0; -- intelsat VI

INTELSAT_5_HOUSEKEEPING_POWER: constant FLOAT := 211.0, -- intelsat V


  TEMPERATURE_AFTER_EQUINOX,

  ABSOLUTE_TEMPERATURE,
  EQUILIBRIUM_TEMPERATURE,
  TIME_MINIMUM_TEMP_ECLIPSE,
HOUSEKEEPING_POWER,

HEAT_RADIATED_TO_SPACE,
TIME_CONSTANT,
MINIMUM_OPERATING_TEMPERATURE,
RADIATOR_AREA,
RADIATOR_HEIGHT,
SOLAR_ARRAY_DIAMETER,

THERMAL_DISSIPATION,
PERCENT_PAYLOAD_POWER_DISSIPATION,
HOUSEKEEPING_PERCENT_POWER_DISSIPATION,
TEMPERATURE_EQUINOX,
TEMPERATURE_EQUINOX_CELCIUS,
X                              : FLOAT ;

STEFAN_BOLTZMANN                        : FLOAT := 5.67E-08;
EFFICIENCY                        : FLOAT := 0.9;
SOLAR_ASPECT_COEFFICIENT_SOLSTICE       : FLOAT := 23.5;
SOLAR_ASPECT_COEFFICIENT_EQUINOX        : FLOAT := 0.0 ;
SOLAR_INTENSITY_SOLSTICE                : FLOAT := 1397.0;  -- W/m^2
SOLAR_INTENSITY_EQUINOX                 : FLOAT := 1362.0;  -- W/m^2
RADIATOR_TEMPERATURE                    : FLOAT := 310.0; -- degrees kelvin
RADIATOR_EMITTANCE_EOL                  : FLOAT := 0.8;
SOLAR_ABSORBTANCE_EOL                   : FLOAT := 0.21;  -- optical solar reflector
SOLAR_EMITTANCE_EOL                     : FLOAT := 0.8;   -- optical solar reflector
BOL_SOLAR_ABSORBTANCE                   : FLOAT := 0.08;  -- optical solar reflector
BOL_SOLAR_EMITTANCE                     . FLOAT := 0.8;   -- optical solar reflector
ABSOLUTE_ZERO                      : FLOAT := 273.0; -- kelvin
NUMBER_THERMAL_EMITTING_FACES           : FLOAT := 2.0; -- # thermal emitting faces
MASS_RADIATOR_PLUS_EQUIPMENT            : FLOAT := 85.0; -- Kgs
PERCENT_PARTIAL_POWER                   : FLOAT := 0.5;  -- %



OKAY                          : BOOLEAN := TRUE;
DRUM_SPINNER                       : BOOLEAN := FALSE;

Y,
y,
N,
n,
TAKE,
CHAR                       : CHARACTER ;

I,
DECISION,
J                       : INTEGER ;

OUTH                       :FILE_TYPE;
```

```
procedure PRINT_HEADER is
  begin
    VIDEO.CLEAR_SCREEN;SET_LINE(1);
    NEW_LINE(2);
    SET_COL(10);
    PUT_LINE("This program walks through a THERMAL CONTROL design for");
    SET_COL(10);
    PUT_LINE("a solar powered geosynchronous satellite.");
    SET_COL(10);
    PUT_LINE("All pertinent data will be saved to a file called THERMAL.DAT");
    NEW_LINE;
  end PRINT_HEADER;

procedure DUAL_SPIN (DRUM_SPINNER : in out BOOLEAN) is
begin
  SET_COL(10);
  PUT_LINE("Is your spacecraft Spin Stabilized ");
  SET_COL(15);
  GET_CHARACTER(char);
  if CHAR = 'Y' or CHAR = 'y' then
    DRUM_SPINNER:=TRUE;
    if DRUM_SPINNER = TRUE then
        VIDEO.CLEAR_SCREEN;SET_LINE(1);
        SET_COL(10);
        PUT_LINE("Satellite is Spin Stabilized");
        NEW_LINE(2);
        PUT_LINE("**********************************************************************");
    end if;
  else
    VIDEO.CLEAR_SCREEN;SET_LINE(1);
    SET_COL(10);
    PUT_LINE("Satellite is Three Axis Stabilized");
    NEW_LINE(2);
    PUT_LINE("**********************************************************************");
  end if;
end DUAL_SPIN;



procedure OPERATING_DATA  (BATTERY_LOAD                    : in out FLOAT;
                           DRUM_SPINNER                    : in out BOOLEAN;
                           RADIATOR_SPECIFIC_HEAT          : in out FLOAT;
                           RADIATOR_EMITTANCE_EOL          : in out FLOAT;
                           MASS_RADIATOR_PLUS_EQUIPMENT    : in out FLOAT;
                           EFFICIENCY                      : in out FLOAT;
                           NUMBER_THERMAL_EMITTING_FACES   : in out FLOAT;
                           PAYLOAD_POWER                   : in out FLOAT;
                           ECLIPSE_TIME                    : in out FLOAT) is
  SPACECRAFT_MASS_BEFORE_APOGEE_BURN,
  SOLAR_ARRAY_LOAD,
  MASS_REFERENCE,
  HOUSEKEEPING_POWER_REFERENCE        : FLOAT;


  REPLACE                        : BOOLEAN := FALSE;
  CHOICE,
  CHANGE,
  INPUT           : INTEGER ;



begin
 if DRUM_SPINNER = FALSE then
  PUT_LINE(OUTH,"SPACECRAFT IS THREE AXIS STABILIZED");
  PUT_LINE(OUTH,"***********************************");
  NEW_LINE(OUTH,1);
 else
  PUT_LINE(OUTH,"SPACECRAFT IS SPIN STABILIZED");
  PUT_LINE(OUTH,"*****************************");
  NEW_LINE(OUTH,1);
 end if;

 SET_COL(10);
 PUT_LINE("Enter the mass of the spacecraft in kilograms");
 SET_COL(10);
 GET_DATA(SPACECRAFT_MASS_BEFORE_APOGEE_BURN);
 VIDEO.CLEAR_SCREEN;SET_LINE(1);
 SET_COL(15);
 PUT("Spacecraft mass before apogee motor burn is  ");
 PUT(SPACECRAFT_MASS_BEFORE_APOGEE_BURN, FORE => 5, AFT => 1, EXP => 0);
 PUT("  kgs");

 PUT(OUTH,"Spacecraft mass before apogee motor burn is  ");
 SET_COL(OUTH,55);
 PUT(OUTH,SPACECRAFT_MASS_BEFORE_APOGEE_BURN, FORE => 5, AFT => 2, EXP => 0);
 PUT(OUTH,"  kgs");
 NEW_LINE(OUTH,1);

 NEW_LINE(2):
 PUT_LINE("********************************************************************");
 NEW_LINE(2);

  MASS_REFERENCE:=INTELSAT_5_REFERENCE;
  HOUSEKEEPING_POWER_REFERENCE:=INTELSAT_5_HOUSEKEEPING_POWER:

-- The mass of the electrical power system is
  SET_COL(10);
  PUT_LINE("Enter the POWER requirements of the Spacecraft in watts.");

  SET_COL(15);
```

```
GET_DATA(PAYLOAD_POWER);
NEW_LINE(2);
SET_COL(15);
VIDEO.CLEAR_SCREEN;
PUT("Payload power requirements are ");
PUT(PAYLOAD_POWER, FORE => 6, AFT => 2, EXP => 0);
PUT(" Watts");

PUT(OUTH,"Payload power requirements are ");
SET_COL(OUTH,55);
PUT(OUTH,PAYLOAD_POWER, FORE => 5, AFT => 2, EXP => 0);
PUT(OUTH," Watts");
NEW_LINE(OUTH,1);

NEW_LINE(2);
PUT_LINE("************************************************************************");
NEW_LINE(2);
SET_COL(5);
PUT_LINE("Choose which satellite you want as your reference for ");
SET_COL(5);
PUT_LINE("housekeeping power and spacecraft mass in kilograms.");
NEW_LINE(2);
PUT_LINE("************************************************************************");
NEW_LINE(1);
PUT_LINE("                '1'          '2'          '3'        ");
SET_COL(5);
PUT_LINE("              Intelsat V    Intelsat VI   Intelsat VII");
SET_COL(5);
PUT_LINE("Mass           1900.0 kgs    2227.0 kgs    3445.0 kgs");
SET_COL(5);
PUT_LINE("Housekeeping    211.0         347.0         613.0 ");
SET_COL(5);
PUT_LINE("Power" );
PUT_LINE("************************************************************************");
SET_COL(5);
PUT_LINE("For an INTELSAT V   reference    enter  integer  '1'  ");
SET_COL(5);
PUT_LINE("For an INTELSAT VI  reference    enter  integer  '2'  ");
SET_COL(5);
PUT_LINE("For an INTELSAT VII reference    enter  integer  '3'  ");
SET_COL(5);
PUT_LINE("For your own reference value's   enter  integer  '4'  ");
SET_COL(5);
PUT_LINE("NO CHANGES  ie intelsat 5 values enter  integer  '5'  ");
GET_INTEGER(CHOICE);

case CHOICE is
  when 1 =>
      MASS_REFERENCE:=INTELSAT_5_REFERENCE;
      HOUSEKEEPING_POWER_REFERENCE:=INTELSAT_5_HOUSEKEEPING_POWER;
      NEW_LINE(OUTH,1);
      PUT(OUTH,"MASS and HOUSEKEEPING  REFERENCE INTELSAT 5");
```

```
          NEW_LINE(OUTH,1);


      when 2=>
          MASS_REFERENCE:=INTELSAT_6_REFERENCE;
          HOUSEKEEPING_POWER_REFERENCE:=INTELSAT_6_HOUSEKEEPING_POWER;
          NEW_LINE(OUTH,1);
          PUT(OUTH,"MASS and HOUSEKEEPING  REFERENCE INTELSAT 6");
          NEW_LINE(OUTH,1);

      when 3=>
          MASS_REFERENCE:=INTELSAT_7_REFERENCE;
          HOUSEKEEPING_POWER_REFERENCE:=INTELSAT_7_HOUSEKEEPING_POWER;
          NEW_LINE(OUTH,1);
          PUT(OUTH,"MASS and HOUSEKEEPING  REFERENCE INTELSAT 7");
          NEW_LINE(OUTH,1);

      when 4=>
          NEW_LINE(2);
          PUT_LINE("**********************************************************************");
          NEW_LINE(2);
          VIDEO.CLEAR_SCREEN;
          PUT("Please enter desired REFERENCE  MASS");
          SET_COL(15);
          GET_DATA(MASS_REFERENCE);
          PUT(OUTH,"INPUT MASS REFERENCE ");
          SET_COL(OUTH,55);
          PUT(OUTH,MASS_REFERENCE,FORE=>4,AFT=>2,EXP=>0);
          NEW_LINE(OUTH,1);
          VIDEO.CLEAR_SCREEN;
          NEW_LINE(2);
          PUT_LINE("**********************************************************************");
          NEW_LINE(2);
          PUT("Please enter desired HOUSEKEEPING POWER reference");
          SET_COL(15);
          GET_DATA(HOUSEKEEPING_POWER_REFERENCE);
          PUT(OUTH,"INPUT HOUSEKEEPING POWER REFERENCE ");
          SET_COL(OUTH,55);
          PUT(OUTH,HOUSEKEEPING_POWER_REFERENCE,FORE=>4,AFT=>2,EXP=>0);
          NEW_LINE(OUTH,1);
          VIDEO.CLEAR_SCREEN;
      when OTHERS =>
          NEW_LINE(2);
          SET_COL(5);
          PUT("Understand INTELSAT V DATA WILL BE USED");
          PUT(OUTH,"MASS and HOUSEKEEPING  REFERENCE INTELSAT 5");
          NEW_LINE(OUTH,1);
    end case;
```

```
HOUSEKEEPING_POWER:=(SPACECRAFT_MASS_BEFORE_APOCEE_BURN
                    /MASS_REFERENCE)
                    *HOUSEKEEPING_POWER_REFERENCE;
VIDEO.CLEAR_SCREEN;
NEW_LINE;
SET_COL(10);
PUT("Housekeeping power is  ");
SET_COL(60);
PUT(HOUSEKEEPING_POWER, FORE => 6, AFT => 2, EXP => 0);



------------------------------------------------------------------



NEW_LINE(3);

PUT_LINE("The Housekeeping Power value will be used in future calculations if");
PUT_LINE("you want to use this value enter a 'y' for YES. If you wish to ");
PUT_LINE("change the value enter a 'n' for NO and the value you enter will ");
put_line("be used in further further calculations");
NEW_LINE(2);
PUT_LINE("*******************************************************************");
NEW_LINE(2);
GET_CHARACTER(CHAR);
if CHAR = 'Y' or CHAR = 'y' then
  VIDEO.CLEAR_SCREEN;
  NEW_LINE(1);
  PUT("Payload power requirements are ");
  PUT(PAYLOAD_POWER, FORE => 6, AFT => 2, EXP => 0);
  PUT("  Watts");
  NEW_LINE(3);
  PUT("Please enter a value for the housekeeping power");
  GET_DATA(HOUSEKEEPING_POWER);
  VIDEO.CLEAR_SCREEN;
  NEW_LINE(2);
  PUT_LINE("*******************************************************************");
  NEW_LINE(2);

  PUT("Housekeeping Power is  ");
  SET_COL(60);
  PUT(HOUSEKEEPING_POWER, FORE => 6, AFT => 2, EXP => 0);
  NEW_LINE(3)·
  PUT("Payload power requirements are ");
  PUT(PAYLOAD_POWER, FORE => 6, AFT => 2, EXP => 0);
  PUT("  Watts");
  NEW_LINE(3);
  STOP;

end if;
```

```
     PUT(OUTH,"Housekeeping power is  ");
     SET_COL(OUTH,55);
     PUT(OUTH,HOUSEKEEPING_POWER, FORE => 5, AFT => 2, EXP => 0);
     NEW_LINE(OUTH,1);

   <<NEW_VALUE>>


     VIDEO.CLEAR_SCREEN;
     PUT_LINE("FOLLOWING IS A LIST OF VARIABLES AND THEIR DEFAULT VALUES");
     PUT_LINE("--------------------------------------------------------");

     PUT("EFFICIENCY                         [1] ");
     SET_COL(56);
     PUT(EFFICIENCY,FORE=>4,AFT=>2,EXP=>0);
     NEW_LINE(1);

     PUT("SOLAR_ASPECT_COEFFICIENT_SOLSTICE        [2]");
     SET_COL(56);
     PUT(SOLAR_ASPECT_COEFFICIENT_SOLSTICE,FORE=>4,AFT=>2,EXP=>0);
     PUT(" degrees");
     NEW_LINE(1);

     PUT("SOLAR_INTENSITY_SOLSTICE               [3]");
     SET_COL(56);
     PUT(SOLAR_INTENSITY_SOLSTICE,FORE=>4,AFT=>2,EXP=>0);
     PUT(" Watts/m^2");
     NEW_LINE(1);

     PUT("SOLAR_INTENSITY_EQUINOX                [4]");
     SET_COL(56);
     PUT(SOLAR_INTENSITY_EQUINOX,FORE=>4,AFT=>2,EXP=>0);
     PUT(" Watts/m^2");
     NEW_LINE(1);

     PUT("RADIATOR_TEMPERATURE                   [5]");
     SET_COL(56);
     PUT(RADIATOR_TEMPERATURE,FORE=>4,AFT=>2,EXP=>0);
     PUT(" kelvin");
     NEW_LINE(1);

     PUT("RADIATOR_EMITTANCE_EOL                 [6]");
     SET_COL(56);
     PUT(RADIATOR_EMITTANCE_EOL,FORE=>4,AFT=>2,EXP=>0);
     NEW_LINE(1);

     PUT("ABSOLUTE_ZERO                        ");
     SET_COL(56);
     PUT(ABSOLUTE_ZERO,FORE=>4,AFT=>2,EXP=>0);
     PUT(" kelvin");
     NEW_LINE(1);
```

```
PUT("TIME_ECLIPSE                    [7]");
SET_COL(56);
PUT(TIME_ECLIPSE,FORE=>4,AFT=>2,EXP=>0);
PUT("  kelvin");
NEW_LINE(1);

PUT("NUMBER_THERMAL_EMITTING_FACES          [8]");
if DRUM_SPINNER = TRUE and REPLACE = FALSE then
  NUMBER_THERMAL_EMITTING_FACES:=1.0;
end if;
SET_COL(56);
PUT(NUMBER_THERMAL_EMITTING_FACES,FORE=>4,AFT=>2,EXP=>0);
NEW_LINE(1);

PUT("MASS_RADIATOR_PLUS_EQUIPMENT           [9]");
SET_COL(56);
PUT(MASS_RADIATOR_PLUS_EQUIPMENT,FORE=>4,AFT=>2,EXP=>0);
PUT("  kgs");
NEW_LINE(1);

PUT("PERCENT_PARTIAL_POWER                 [10]");
SET_COL(56);
PUT(PERCENT_PARTIAL_POWER*100.0,FORE=>4,AFT=>2,EXP=>0);
PUT("  %");
NEW_LINE(1);

PUT("RADIATOR_SPECIFIC_HEAT                [11]");
SET_COL(56);
PUT(RADIATOR_SPECIFIC_HEAT,FORE=>4,AFT=>2,EXP=>0);
PUT("  (watts*sec)\(kg*Kelvin)");
NEW_LINE(1);

if REPLACE = FALSE then
  CHAR := N;
  PUT_LINE("If you desire to change any of the listed values please enter ");
  GET_CHARACTER(CHAR);
  if CHAR = 'Y' or CHAR = 'y' then
      CHAR := N;
      PUT_LINE("Enter number corresponding to value you wish to change.");
      set_col(10);
      GET_INTEGER(CHANGE);
      VIDEO.CLEAR_SCREEN;
  else
      VIDEO.CLEAR_SCREEN;
      goto KEEP_VALUES;
  end if;
elsif REPLACE = TRUE then
  PUT_LINE("Enter number corresponding to value you wish to change.");
  set_col(10);
  GET_INTEGER(CHANGE);
  VIDEO.CLEAR_SCREEN;
end if;
```

```
VIDEO.CLEAR_SCREEN;
case CHANGE is
    when 1 = >
      PUT("Please enter a value for Efficiency (usually around 0.9)");
      SET_COL(15);
      GET_DATA(EFFICIENCY);
      NEW_LINE(2);
      PUT_LINE("************************************************************************");
      NEW_LINE(2);
      PUT("EFFICIENCY");
      SET_COL(50);
      PUT(EFFICIENCY,FORE= >4,AFT= >2,EXP= >0);
      NEW_LINE(1);

    when 2 = >
      PUT("Please enter a value for Solar Aspect (usually 23.5 degrees)");
      SET_COL(15);
      GET_DATA(SOLAR_ASPECT_COEFFICIENT_SOLSTICE);
      NEW_LINE(2);
      PUT_LINE("************************************************************************");
      NEW_LINE(2);
      PUT("SOLAR_ASPECT_COEFFICIENT_SOLSTICE");
      SET_COL(50);
      PUT(SOLAR_ASPECT_COEFFICIENT_SOLSTICE,FORE= >4,AFT= >2,EXP= >0);
      PUT("  degrees");
      NEW_LINE(1);

    when 3 = >
      PUT("Please enter a value for Solar Intensity at Solstice (1397.0)");
      SET_COL(15);
      GET_DATA(SOLAR_INTENSITY_SOLSTICE);
      NEW_LINE(2);
      PUT_LINE("************************************************************************");
      NEW_LINE(2);
      PUT("SOLAR_INTENSITY_SOLSTICE");
      SET_COL(50);
      PUT(SOLAR_INTENSITY_SOLSTICE,FORE= >4,AFT= >2,EXP= >0);
      PUT("  Watts/m^2");
      NEW_LINE(1);

    when 4 = >
      PUT("Please enter a value for Solar Intensity at Equinox (1362.0)");
      SET_COL(15);
      GET_DATA(SOLAR_INTENSITY_EQUINOX);
      NEW_LINE(2);
      PUT_LINE("************************************************************************");
      NEW_LINE(2);
      PUT("SOLAR_INTENSITY_EQUINOX");
      SET_COL(50);
```

```
  PUT(SOLAR_INTENSITY_EQUINOX,FORE=>4,AFT=>2,EXP=>0);
  PUT(" Watts/m^2");
  NEW_LINE(1);

when 5 =>
  PUT("Please enter a value for Radiator Temperature (310.0)");
  SET_COL(15);
  GET_DATA(RADIATOR_TEMPERATURE);
  NEW_LINE(2);
  PUT_LINE("**********************************************************************");
  NEW_LINE(2);
  PUT("RADIATOR_TEMPERATURE");
  SET_COL(50);
  PUT(RADIATOR_TEMPERATURE,FORE=>4,AFT=>2,EXP=>0);
  PUT(" kelvin");
  NEW_LINE(1);

when 6 =>
  PUT("Please enter a value for Radiator Emittance  (0.8)");
  SET_COL(15);
  GET_DATA(RADIATOR_EMITTANCE_EOL);
  NEW_LINE(2);
  PUT_LINE("**********************************************************************");
  NEW_LINE(2);
  PUT("RADIATOR_EMITTANCE_EOL");
  SET_COL(50);
  PUT(RADIATOR_EMITTANCE_EOL,FORE=>4,AFT=>2,EXP=>0);
  NEW_LINE(1);

when 7 =>
  PUT("Please enter a value for Eclipse Time (72.0 minutes)");
  SET_COL(15);
  GET_DATA(TIME_ECLIPSE);
  NEW_LINE(2);
  PUT_LINE("**********************************************************************");
  NEW_LINE(2);
  PUT("TIME_ECLIPSE");
  SET_COL(50);
  PUT(TIME_ECLIPSE,FORE=>4,AFT=>2,EXP=>0);
  PUT(" kelvin");
  NEW_LINE(1);

when 8 =>
  PUT("Please enter a value for Emitting Faces (2.0)");
  SET_COL(15);
  GET_DATA(NUMBER_THERMAL_EMITTING_FACES);
  NEW_LINE(2);
  PUT_LINE("**********************************************************************");
  NEW_LINE(2);
  PUT("NUMBER_THERMAL_EMITTING_FACES");
  SET_COL(50);
  PUT(NUMBER_THERMAL_EMITTING_FACES,FORE=>4,AFT=>2,EXP=>0);
```

```
     NEW_LINE(1);

   when 9 =>
     PUT("Please enter a value for Mass of Radiator plus equipment (85.0)");
     SET_COL(15);
     GET_DATA(MASS_RADIATOR_PLUS_EQUIPMENT);
     NEW_LINE(2);
     PUT_LINE("***********************************************************************");
     NEW_LINE(2);
     PUT("MASS_RADIATOR_PLUS_EQUIPMENT");
     SET_COL(50);
     PUT(MASS_RADIATOR_PLUS_EQUIPMENT,FORE=>4,AFT=>2,EXP=>0);
     PUT(" kgs");
     NEW_LINE(1);

   when 10 =>
     PUT("Please enter a value for Percent Partial Power (0.5)");
     SET_COL(15);
     GET_DATA(PERCENT_PARTIAL_POWER);
     NEW_LINE(2);
     PUT_LINE("***********************************************************************");
     NEW_LINE(2);
     PUT("PERCENT_PARTIAL_POWER");
     SET_COL(50);
     PUT(PERCENT_PARTIAL_POWER*100.0,FORE=>4,AFT=>2,EXP=>0);
     PUT(" %");
     NEW_LINE(1);

   when 11 =>
     PUT("Please enter a value for Radiator Specific Heat");
     SET_COL(15);
     GET_DATA(RADIATOR_SPECIFIC_HEAT);
     NEW_LINE(2);
     PUT_LINE("***********************************************************************");
     NEW_LINE(2);
     PUT("RADIATOR_SPECIFIC_HEAT");
     SET_COL(50);
     PUT(RADIATOR_SPECIFIC_HEAT,FORE=>4,AFT=>2,EXP=>0);
     PUT(" (watts*sec)\(kg*Kelvin) ");
     NEW_LINE(1);

   when others =>
     video.clear_screen;
     PUT_LINE("UNDERSTAND NO MORE CHANGES PLEASE ENTER AN 'N' FOR NEXT
QUESTION");
     PUT_LINE("IF YOU STILL DESIRE TO MAKE CHANGES ENTER 'Y'");
     NEW_LINE(3);

   end case;
   NEW_LINE(2);
   PUT_LINE("***********************************************************************");
   NEW_LINE(2);
```

```
        NEW_LINE(2);
        CHAR := N;
        PUT_LINE("If you wish to change a value please enter a 'y' for YES");
        PUT_LINE("otherwise enter a 'n' for NO ");
        REPLACE:=FALSE;
        GET_CHARACTER(CHAR);
        if CHAR = 'Y' or CHAR = 'y' then
          CHAR := N;
          REPLACE:=TRUE;
          VIDEO.CLEAR_SCREEN;
          goto NEW_VALUE;

        else
          VIDEO.CLEAR_SCREEN;
          goto KEEP_VALUES;
          NEW_LINE(3);
        end if;
    STOP;

<<KEEP_VALUES>>

  PUT(OUTH,"EFFICIENCY");
  SET_COL(OUTH,55);
  PUT(OUTH,EFFICIENCY,FORE=>5,AFT=>2,EXP=>0);
  NEW_LINE(OUTH,1);

  PUT(OUTH,"SOLAR ANGLE SOLSTICE");
  SET_COL(OUTH,55);
  PUT(OUTH,SOLAR_ASPECT_COEFFICIENT_SOLSTICE,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH," degrees");
  NEW_LINE(OUTH,1);

  PUT(OUTH,"SOLAR INTENSITY SOLSTICE");
  SET_COL(OUTH,55);
  PUT(OUTH,SOLAR_INTENSITY_SOLSTICE,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH," Watts/m^2");
  NEW_LINE(OUTH,1);

  PUT(OUTH,"SOLAR INTENSITY EQUINOX");
  SET_COL(OUTH,55);
  PUT(OUTH,SOLAR_INTENSITY_EQUINOX,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH," Watts/m^2");
  NEW_LINE(OUTH,1);
  PUT(OUTH,"RADIATOR TEMPERATURE");
  SET_COL(OUTH,55);
  PUT(OUTH,RADIATOR_TEMPERATURE,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH," kelvin");
  NEW_LINE(OUTH,1);

  PUT(OUTH,"RADIATOR EMITTANCE");
  SET_COL(OUTH,55);
  PUT(OUTH,RADIATOR_EMITTANCE_EOL,FORE=>5,AFT=>2,EXP=>0);
  NEW_LINE(OUTH,1);

  PUT(OUTH,"ABSOLUTE ZERO");
  SET_COL(OUTH,55);
  PUT(OUTH,ABSOLUTE_ZERO,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH," kelvin");
  NEW_LINE(OUTH,1);

  PUT(OUTH,"TIME ECLIPSE");
  SET_COL(OUTH,55);
  PUT(OUTH,TIME_ECLIPSE,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH," kelvin");
  NEW_LINE(OUTH,1);

  PUT(OUTH,"NUMBER THERMAL EMITTING FACES");
  SET_COL(OUTH,55);
  PUT(OUTH,NUMBER_THERMAL_EMITTING_FACES,FORE=>5,AFT=>2,EXP=>0);
  NEW_LINE(OUTH,1);

  PUT(OUTH,"MASS RADIATOR PLUS EQUIPMENT");
  SET_COL(OUTH,55);
  PUT(OUTH,MASS_RADIATOR_PLUS_EQUIPMENT,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH," kgs");
  NEW_LINE(OUTH,1);

  PUT(OUTH,"PERCENT PARTIAL POWER");
  SET_COL(OUTH,55);
  PUT(OUTH,PERCENT_PARTIAL_POWER*100.0,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH," %");
  NEW_LINE(OUTH,1);

  PUT(OUTH,"RADIATOR SPECIFIC HEAT");
  SET_COL(OUTH,55);
  PUT(OUTH,RADIATOR_SPECIFIC_HEAT,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH," (watts*sec)\(kg*Kelvin)");
  NEW_LINE(OUTH,1);



end OPERATING_DATA;

procedure HEAT    (BATTERY_LOAD                 : in out FLOAT;
                   DRUM_SPINNER                 : in out BOOLEAN;
                   RADIATOR_AREA                : in out FLOAT;
                   PAYLOAD_POWER                : in out FLOAT;
                   TEMPERATURE_AFTER_EQUINOX    : in out FLOAT;
                   THERMAL_DISSIPATION          : in out FLOAT;
                   ECLIPSE_TIME                 : in out FLOAT) is

TIME_ECLIPSE        : FLOAT :=72.0;
```

```
ECLIPSE_TIME_MINUTES : constant FLOAT := 72.0; -- eclipse time in minutes

TIME_CONSTANT,
TIME_CONSTANT_MINUTES,
TIME_ECLIPSE_ONE,
TIME_ECLIPSE_TWO,
EQUINOX_TEMP_ONE,
EQUINOX_TEMP_TWO,
CONST,


SOLAR_ABSORBTANCE_BOL,
SOLAR_ABSORBTANCE_EOL,
RADIATOR_EMITTANCE_EOL,
EMITTANCE_BOL                 : FLOAT ;

begin
  VIDEO.CLEAR_SCREEN;
  PUT_LINE("  Thermal properties of surfaces are listed in the text");
  PUT_LINE("Design of Geosynchronous Spacecraft by Brij  Agrawal");
  PUT_LINE("in Table 5.3  pg. 275");
  PUT_LINE("the properties listed are Solar Absorbtance for EOL and BOL");
  PUT_LINE("as well as Emittance for EOL and  BOL");
  PUT_LINE("The following surfaces are listed and can be used for thermal");
  PUT_LINE("radiation surfaces in your calculations.");
  NEW_LINE(1);
  PUT_LINE("  Surface                    Typical Application ");
  PUT_LINE("--------------------------------------------------------");
  PUT_LINE("  Black Paint                 Interior Structure");
  PUT_LINE("  White Paint                 Antenna Reflector");
  PUT_LINE("  Optical Solar Reflector(OSR)  North & South Panel Reflectors");
  PUT_LINE("  Aluminized Kapton             Thermal Insulation");
  PUT_LINE("  Tiodized Titanium             Apogee Motor Thermal Shield");
  PUT_LINE("  Aluminum ,");
  PUT_LINE("     aluminum tape             Propellent Insulation");
  PUT_LINE("     deposited aluminum");
  PUT_LINE("  Anodized Aluminum             Interior Structure");
  PUT_LINE("  Solar Cells                 Solar Panels");
  STOP;
  VIDEO.CLEAR_SCREEN;
  PUT_LINE("Please enter your choice for thermal emitting surface");
  PUT_LINE("to be used for calculations  try (OSR) CHOICE [3] first ");
  PUT_LINE("------------------------------------------------------------");
  PUT_LINE("SURFACE          CHOICE    SOLAR ABSORBTANCE      EMITTANCE");
  PUT_LINE("------------------------------------------------------------");
  PUT_LINE("                          BOL        EOL        BOL      EOL");
  PUT_LINE("------------------------------------------------------------");
  PUT_LINE("Black Paint         [1]     0.9        0.9        0.9     0.9");
  PUT_LINE("White Paint         [2]     0.2        0.6        0.9     0.9");
  PUT_LINE("Optical Solar Ref.  [3]     0.08       0.21       0.8     0.8");
  PUT_LINE("Graphite Epoxy      [4]     0.84       0.84       0.85    0.85");
  PUT_LINE("Aluminized Kapton   [5]     0.35       0.50       0.6     0.6");
PUT_LINE("Tiodized Titanium  [6]      0.6         0.6         0.6       0.6");
PUT_LINE("Aluminum Depos/Tape [7]      0.12        0.18        0.06      0.06");
PUT_LINE("Anodized Aluminum  [8]      0.2         0.6         0.8       0.8 ");
PUT_LINE("Solar Cells         [9]      0.7         0.7         0.82      0.82");
PUT_LINE("YOUR VALUES         [10] ");
GET_INTEGER(DECISION);
video.clear_screen;
case DECISION is

  when 1=>
      SOLAR_ABSORBTANCE_BOL := 0.9  ;
      SOLAR_ABSORBTANCE_EOL := 0.9  ;
      RADIATOR_EMITTANCE_EOL:= 0.9  ;
      EMITTANCE_BOL         := 0.9  ;
      PUT_LINE(OUTH,"RADIATOR MATERIAL IS BLACK PAINT");
  when 2=>
      SOLAR_ABSORBTANCE_BOL := 0.2  ;
      SOLAR_ABSORBTANCE_EOL := 0.6  ;
      RADIATOR_EMITTANCE_EOL:= 0.9  ;
      EMITTANCE_BOL         := 0.9  ;
      PUT_LINE(OUTH,"RADIATOR MATERIAL IS WHITE PAINT");

  when 3=>
      SOLAR_ABSORBTANCE_BOL := 0.08 ;
      SOLAR_ABSORBTANCE_EOL := 0.21 ;
      RADIATOR_EMITTANCE_EOL:= 0.8  ;
      EMITTANCE_BOL         := 0.8  ;
      PUT_LINE(OUTH,"RADIATOR MATERIAL IS OPTICAL SOLAR REFLECTOR");

  when 4=>
      SOLAR_ABSORBTANCE_BOL := [illegible].84 ;
      SOLAR_ABSORBTANCE_EOL := 0.84 ;
      RADIATOR_EMITTANCE_EOL:= 0.85 ;
      EMITTANCE_BOL         := 0.85 ;
      PUT_LINE(OUTH,"RADIATOR MATERIAL IS GRAPHITE EPOXY");

  when 5=>
      SOLAR_ABSORBTANCE_BOL := 0.35 ;
      SOLAR_ABSORBTANCE_EOL := 0.50 ;
      RADIATOR_EMITTANCE_EOL:= 0.6  ;
      EMITTANCE_BOL         := 0.6  ;
      PUT_LINE(OUTH,"RADIATOR MATERIAL IS ALUMINIZED KAPTON");
  when 6=>
      SOLAR_ABSORBTANCE_BOL := 0.6  ;
      SOLAR_ABSORBTANCE_EOL := 0.6  ;
      RADIATOR_EMITTANCE_EOL:= 0.6  ;
      EMITTANCE_BOL         := 0.6  ;
      PUT_LINE(OUTH,"RADIATOR MATERIAL IS TIODIZED TITANIUM");
  when 7=>
      SOLAR_ABSORBTANCE_BOL := 0.12 ;
      SOLAR_ABSORBTANCE_EOL := 0.18 ;
      RADIATOR_EMITTANCE_EOL:= 0.06 ;
```

```
    EMITTANCE_BOL         := 0.06  ;
    PUT_LINE(OUTH,"RADIATOR MATERIAL IS ALUMINUM DEPOSITS OR TAPE");

when 8=>
    SOLAR_ABSORBTANCE_BOL := 0.2   ;
    SOLAR_ABSORBTANCE_EOL := 0.6   ;
    RADIATOR_EMITTANCE_EOL:= 0.8   ;
    EMITTANCE_BOL         := 0.8   ;
    PUT_LINE(OUTH,"RADIATOR MATERIAL IS ANODIZED ALUMINUM");

when 9=>
    SOLAR_ABSORBTANCE_BOL := 0.7   ;
    SOLAR_ABSORBTANCE_EOL := 0.7   ;
    RADIATOR_EMITTANCE_EOL:= 0.82  ;
    EMITTANCE_BOL         := 0.82  ;
    PUT_LINE(OUTH,"RADIATOR MATERIAL IS SOLAR CELLS");

when 10=>
    VIDEO.CLEAR_SCREEN;
    PUT("Please enter value for SOLAR ABSORBTANCE at BOL");
    SET_COL(15);
    GET_DATA(SOLAR_ABSORBTANCE_BOL);
    NEW_LINE(2);
    PUT_LINE("********************************************************************");
    NEW_LINE(2);
    VIDEO.CLEAR_SCREEN;
    PUT("Please enter value for SOLAR ABSORBTANCE at EOL");
    SET_COL(15);
    GET_DATA(SOLAR_ABSORBTANCE_EOL);
    NEW_LINE(2);
    PUT_LINE("********************************************************************");
    NEW_LINE(2);
    VIDEO.CLEAR_SCREEN;
    PUT("Please enter value for EMITTANCE at BOL");
    SET_COL(15);
    GET_DATA(EMITTANCE_BOL);
    NEW_LINE(2);
    PUT_LINE("********************************************************************");
    NEW_LINE(2);
    VIDEO.CLEAR_SCREEN;
    NEW_LINE(2);
    PUT_LINE("********************************************************************");
    NEW_LINE(2);
    VIDEO.CLEAR_SCREEN;
    PUT("Please enter value for EMITTANCE at EOL");
    SET_COL(15);
    GET_DATA(RADIATOR_EMITTANCE_EOL);
    NEW_LINE(2);
    PUT_LINE("********************************************************************");
    NEW_LINE(2);
    VIDEO.CLEAR_SCREEN ;
    when others=>
        SOLAR_ABSORBTANCE_BOL:= 0.08  ;
        SOLAR_ABSORBTANCE_EOL:= 0.21  ;
        RADIATOR_EMITTANCE_EOL:= 0.8   ;
        EMITTANCE_BOL        := 0.8   ;
        NEW_LINE(2);
        PUT(" Values for absorbtance and emittance are for Optical Solar Reflector (OSR)");
        NEW_LINE(1);

end CASE;


  PUT_LINE("Values for absorbtance and emittance are: ");
  NEW_LINE(2);
  PUT("SOLAR ABSORBTANCE BOL ");
  SET_COL(60);
  PUT(SOLAR_ABSORBTANCE_BOL,FORE=>1,AFT=>2,EXP=>0);

  NEW_LINE(2);
  PUT("SOLAR ABSORBTANCE EOL ");
  SET_COL(60);
  PUT(SOLAR_ABSORBTANCE_EOL,FORE=>1,AFT=>2,EXP=>0);

  NEW_LINE(2);
  PUT("RADIATOR EMITTANCE BOL ");
  SET_COL(60);
  PUT(EMITTANCE_BOL,FORE=>1,AFT=>2,EXP=>0);

  NEW_LINE(2);
  PUT("RADIATOR EMITTANCE EOL ");
  SET_COL(60);
  PUT(RADIATOR_EMITTANCE_EOL,FORE=>1,AFT=>2,EXP=>0);

  PUT(OUTH,"Values for absorbtance and emittance are: ");
  NEW_LINE(OUTH,1);
  PUT(OUTH,"SOLAR ABSORBTANCE BOL ");
  SET_COL(OUTH,55);
  PUT(OUTH,SOLAR_ABSORBTANCE_BOL,FORE=>5,AFT=>2,EXP=>0);
  NEW_LINE(OUTH,1);

  PUT(OUTH,"SOLAR ABSORBTANCE EOL ");
  SET_COL(OUTH,55);
  PUT(OUTH,SOLAR_ABSORBTANCE_EOL,FORE=>5,AFT=>2,EXP=>0);
  NEW_LINE(OUTH,1);

  PUT(OUTH,"EMITTANCE BOL ");
  SET_COL(OUTH,55);
  PUT(OUTH,EMITTANCE_BOL,FORE=>5,AFT=>2,EXP=>0);
  NEW_LINE(OUTH,1);

  PUT(OUTH,"EMITTANCE EOL ");
  SET_COL(OUTH,55);
  PUT(OUTH,RADIATOR_EMITTANCE_EOL,FORE=>5,AFT=>2,EXP=>0);
  NEW_LINE(OUTH,2);

  NEW_LINE(2);
  PUT_LINE("*****************************************************************");
  NEW_LINE(2);

  STOP;
  VIDEO.CLEAR_SCREEN;

<<TOOBIG>>
  PUT_LINE("Please enter the PERCENT PAYLOAD POWER that must be ");
  PUT_LINE("dissipated as heat Example:       Enter 59%  as   '0.59' ");
  NEW_LINE(2);
  PUT_LINE("*****************************************************************");
  NEW_LINE(2);
  SET_COL(10);
  GET_DATA(PERCENT_PAYLOAD_POWER_DISSIPATION);
  if PERCENT_PAYLOAD_POWER_DISSIPATION > 1.0 then
    PUT_LINE("*****************************************************************");
    VIDEO.CLEAR_SCREEN;
    NEW_LINE(2);
    PUT_LINE("Please use a value less then 1.0");
    NEW_LINE(2);
    PUT_LINE("*****************************************************************");
    goto TOOBIG;
  end if;
  NEW_LINE(2);
  PUT("Percent Payload Power Dissipated as Heat is   ");
  SET_COL(50);
  PUT(PERCENT_PAYLOAD_POWER_DISSIPATION*100.0, FORE=>2,AFT=>2,EXP=>0);
  PUT(" %");
  NEW_LINE(2);

  PUT(OUTH,"Percent Payload Power Dissipated as Heat is   ");
  SET_COL(OUTH,55);
  PUT(OUTH,PERCENT_PAYLOAD_POWER_DISSIPATION*100.0, FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH," %");
  NEW_LINE(OUTH,1);


<<TOOBIGH>>
  PUT_LINE("Please enter the PERCENT OF HOUSEKEEPING POWER that must be ");
  PUT_LINE("dissipated as heat Example:       Enter 23%  as   '0.23' ");
  SET_COL(10);
  GET_DATA(HOUSEKEEPING_PERCENT_POWER_DISSIPATION);
  if HOUSEKEEPING_PERCENT_POWER_DISSIPATION > 1.0 then
    VIDEO.CLEAR_SCREEN;
    PUT_LINE("*****************************************************************");
    NEW_LINE(2);
    PUT_LINE("Please use a value less then 1.0");
    NEW_LINE(2);
```

```
      goto TOOBIGH;
    end if;
    VIDEO.CLEAR_SCREEN;
    NEW_LINE(2);
    PUT("Percent Payload Power Dissipated as Heat is ");
    SET_COL(50);
    PUT(PERCENT_PAYLOAD_POWER_DISSIPATION*100.0, FORE=>2,AFT=>2,EXP=>0);
    PUT(" %");
    NEW_LINE(2);
    PUT("Percent Housekeeping Power Dissipated as Heat is ");
    SET_COL(50);
    PUT(HOUSEKEEPING_PERCENT_POWER_DISSIPATION*100.0, FORE=>2,AFT=>2,EXP=>0);
    PUT(" %");
    PUT(OUTH,"Percent Housekeeping Power Dissipated as Heat is ");
    SET_COL(OUTH,55);
    PUT(OUTH,HOUSEKEEPING_PERCENT_POWER_DISSIPATION*100.0,  FORE=>5,AFT=>2,EXP=>0);
    PUT(OUTH," %");
    NEW_LINE(OUTH,1);
    NEW_LINE(2);
    PUT_LINE("*****************************************************************");
    NEW_LINE(2);
    STOP;
    -- Thermal Dissipation calculation

  if DRUM_SPINNER = FALSE then

    THERMAL_DISSIPATION:=(1.0/NUMBER_THERMAL_EMITTING_FACES)
                *(PAYLOAD_POWER*PERCENT_PAYLOAD_POWER_DISSIPATION
                +HOUSEKEEPING_POWER*HOUSEKEEPING_PERCENT_POWER_DISSIPATION);
    NEW_LINE(2);
    PUT("Thermal Dissipation is ");
    SET_COL(60);
    PUT(THERMAL_DISSIPATION,FORE=>5,AFT=>2,EXP=>0);
    PUT("  Watts");

    PUT(OUTH,"Thermal Dissipation is ");
    SET_COL(OUTH,55);
    PUT(OUTH,THERMAL_DISSIPATION,FORE=>5,AFT=>2,EXP=>0);
    PUT(OUTH," Watts");
    NEW_LINE(OUTH,1);

    -- Radiator Area calculation

    RADIATOR_AREA:=THERMAL_DISSIPATION/((RADIATOR_EMITTANCE_EOL*STEFAN_BOLTZMANN
          *RADIATOR_TEMPERATURE**4.0*EFFICIENCY)-(SOLAR_ABSORBTANCE_EOL
          *SOLAR_INTENSITY_SOLSTICE
          *SIN(SOLAR_ASPECT_COEFFICIENT_SOLSTICE*PI/180.0)));
    NEW_LINE(2);
    PUT("Radiator Area is ");
    SET_COL(60);
    PUT(RADIATOR_AREA,FORE=>5,AFT=>3,EXP=>0);
    PUT("  meters^2");
  NEW_LINE(1);

  PUT(OUTH,"Radiator Area is ");
  SET_COL(OUTH,55);
  PUT(OUTH,RADIATOR_AREA,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH," meters^2");
  NEW_LINE(OUTH,1);

  PUT_LINE("**************************************************************");
  NEW_LINE(2);

  -- temperature during equinox calculation


  STOP;
  VIDEO.CLEAR_SCREEN;

  TEMPERATURE_EQUINOX:=(THERMAL_DISSIPATION

/(RADIATOR_EMITTANCE_EOL*STEFAN_BOLTZMANN*EFFICIENCY*RADIATOR_AREA))**0.25;
  NEW_LINE(2);
  PUT("Temperature at Equinox is Full Power");
  SET_COL(60);
  PUT(TEMPERATURE_EQUINOX,FORE=>5,AFT=>2,EXP=>0);
  PUT(" kelvin");

  PUT(OUTH,"Temperature at Equinox is Full Power");
  SET_COL(OUTH,55);
  PUT(OUTH,TEMPERATURE_EQUINOX,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH," kelvin");
  NEW_LINE(OUTH,1);

  TEMPERATURE_EQUINOX_CELCIUS:= TEMPERATURE_EQUINOX-273.15;
  NEW_LINE(2);
  PUT("Temperature at Equinox (celcius)is ");
  SET_COL(60);
  PUT(TEMPERATURE_EQUINOX_CELCIUS,FORE=>5,AFT=>2,EXP=>0);
  PUT(" celcius");

  PUT(OUTH,"Temperature at Equinox (celcius) Full Power is ");
  SET_COL(OUTH,55);
  PUT(OUTH,TEMPERATURE_EQUINOX_CELCIUS,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH," celcius");
  NEW_LINE(OUTH,2);

  if TEMPERATURE_EQUINOX_CELCIUS >= 5.0
     and TEMPERATURE_EQUINOX_CELCIUS <= 37.0 then
    NEW_LINE(2);
    PUT_LINE("Temperature during equinox non eclipse period is within ");
    PUT_LINE("the prescribed limits of 5 to 37 degrees celcius.  No ");
    PUT_LINE("auxiliary heating is required! ");
    PUT_LINE(OUTH,"Temperature during equinox non eclipse period is within ");
  PUT_LINE(OUTH,"the prescribed limits of 5 to 37 degrees celcius.  No ");
  PUT_LINE(OUTH,"auxiliary heating is required! ");
  NEW_LINE(OUTH,1);
elsif TEMPERATURE_EQUINOX_CELCIUS < 5.0 then
  NEW_LINE(2);
  PUT_LINE(" Temperature is to low auxiliary heaters will be required");
  PUT_LINE(OUTH,"Temperature is to low, auxiliary heaters will be required");
  NEW_LINE(OUTH,2);
end if;
NEW_LINE(2);
PUT_LINE("********************************************************************");
NEW_LINE(3);


STOP;
VIDEO.CLEAR_SCREEN;

-- Equilibrium Temperature calculation

NEW_LINE(2);
PUT_LINE("When batteries provide partial power, heat dissipation  ");
PUT_LINE("and equilibrium temperature during eclipse is  ");
new_line(2);

EQUILIBRIUM_TEMPERATURE:=((THERMAL_DISSIPATION*PERCENT_PARTIAL_POWER)
              /(RADIATOR_EMITTANCE_EOL*STEFAN_BOLTZMANN*RADIATOR_AREA))**0.25;

--------------------


NEW_LINE(1);
SET_COL(45);
PUT(EQUILIBRIUM_TEMPERATURE,FORE=>5,AFT=>2,EXP=>0);
PUT("  degrees kelvin");

PUT(OUTH,"Equilibrium Temperature Partial Power");
SET_COL(OUTH,55);
PUT(OUTH,EQUILIBRIUM_TEMPERATURE,FORE=>5,AFT=>2,EXP=>0);
PUT(OUTH,"  degrees kelvin");
NEW_LINE(OUTH,1);

new_line(2);
PUT_LINE("********************************************************************");
NEW_LINE(3);

-- Time Constant Calculation
STOP;
VIDEO.CLEAR_SCREEN;

TIME_CONSTANT:=(MASS_RADIATOR_PLUS_EQUIPMENT*RADIATOR_SPECIFIC_HEAT)
              /(4.0*RADIATOR_EMITTANCE_EOL*STEFAN_BOLTZMANN
              *RADIATOR_AREA*EQUILIBRIUM_TEMPERATURE**3.0);
  NEW_LINE(2);
  PUT("Time Constant is ");
  SET_COL(50);
  PUT(TIME_CONSTANT,FORE=>7,AFT=>0,EXP=>0);
  PUT("  seconds");
  TIME_CONSTANT_MINUTES:=TIME_CONSTANT/60.0;
  NEW_LINE(2);
  PUT("Time Constant Minutes is ");
  SET_COL(50);
  PUT(TIME_CONSTANT_MINUTES,FORE=>5,AFT=>2,EXP=>0);
  PUT("  minutes");

  PUT(OUTH,"Time Constant is ");
  SET_COL(OUTH,55);
  PUT(OUTH,TIME_CONSTANT,FORE=>7,AFT=>0,EXP=>0);
  PUT(OUTH,"  seconds");
  NEW_LINE(OUTH,1);
  PUT(OUTH,"Time Constant Minutes is ");
  SET_COL(50);
  PUT(OUTH,TIME_CONSTANT_MINUTES,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH,"  minutes");
  NEW_LINE(OUTH,1);

  NEW_LINE(2);
  PUT_LINE("****************************************************************");
  NEW_LINE(2);
  STOP;
  VIDEO.CLEAR_SCREEN;

elsif DRUM_SPINNER = TRUE then
  NUMBER_THERMAL_EMITTING_FACES:=1.0;
  THERMAL_DISSIPATION:=(PAYLOAD_POWER*PERCENT_PAYLOAD_POWER_DISSIPATION
             +HOUSEKEEPING_POWER*HOUSEKEEPING_PERCENT_POWER_DISSIPATION);
  NEW_LINE(2);
  PUT("Thermal Dissipation is ");
  SET_COL(60);
  PUT(THERMAL_DISSIPATION,FORE=>5,AFT=>2,EXP=>0);
  PUT("  Watts");

  PUT(OUTH,"Thermal Dissipation is ");
  SET_COL(OUTH,55);
  PUT(OUTH,THERMAL_DISSIPATION,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH,"  Watts");
  NEW_LINE(OUTH,1);


  -- Radiator Area calculation
--  RADIATOR_HEIGHT,
--  SOLAR_ARRAY_DIAMETER,
  NEW_LINE(2);
  PUT_LINE("****************************************************************");
  NEW_LINE(2);
```

```
PUT("Please enter Solar Array Diameter in meters");
NEW_LINE(1);
SET_COL(15);
GET_DATA(SOLAR_ARRAY_DIAMETER);
VIDEO.CLEAR_SCREEN;
NEW_LINE(1);
PUT("Solar Array Diameter is ");
SET_COL(55);
PUT(SOLAR_ARRAY_DIAMETER,FORE=>5,AFT=>4,EXP=>0);
PUT(" meters");


PUT(OUTH,"Solar Array Diameter is ");
SET_COL(OUTH,54);
PUT(OUTH,SOLAR_ARRAY_DIAMETER,FORE=>5,AFT=>3,EXP=>0);
PUT(OUTH," meters");
NEW_LINE(OUTH,1);


RADIATOR_HEIGHT:=THERMAL_DISSIPATION/(SOLAR_ARRAY_DIAMETER
        *((PI*RADIATOR_EMITTANCE_EOL*STEFAN_BOLTZMANN
        *RADIATOR_TEMPERATURE**4.0*EFFICIENCY)-(SOLAR_ABSORBTANCE_EOL
        *SOLAR_INTENSITY_EQUINOX
        *COS(SOLAR_ASPECT_COEFFICIENT_EQUINOX*PI/180.0))));
NEW_LINE(2);

PUT("Radiator Height is ");
SET_COL(55);
PUT(RADIATOR_HEIGHT,FORE=>5,AFT=>4,EXP=>0);
PUT(" meters");

PUT(OUTH,"Radiator Height is ");
SET_COL(OUTH,55);
PUT(OUTH,RADIATOR_HEIGHT,FORE=>5,AFT=>2,EXP=>0);
PUT(OUTH," meters");
NEW_LINE(OUTH,1);

RADIATOR_AREA:=RADIATOR_HEIGHT*PI*(SOLAR_ARRAY_DIAMETER/2.0)**2.0;
new_line(1);
PUT("Radiator Area (Spin Stabilized) is ");
SET_COL(55);
PUT(RADIATOR_AREA,FORE=>5,AFT=>4,EXP=>0);
PUT(" meters^2");

PUT(OUTH,"Radiator Area is ");
SET_COL(OUTH,55);
PUT(OUTH,RADIATOR_AREA,FORE=>5,AFT=>2,EXP=>0);
PUT(OUTH," meters^2");
NEW_LINE(OUTH,1);
NEW_LINE(2);
PUT_LINE("******************************************************************");
NEW_LINE(2);

-- temperature during equinox calculation

PUT("Please enter a value for Radiator Efficiency (usually around 0.9)");
NEW_LINE(2);
GET_DATA(EFFICIENCY);
NEW_LINE(2);
PUT_LINE("******************************************************************");
NEW_LINE(2);
PUT("EFFICIENCY");
NEW_LINE(2);
SET_COL(50);
PUT(EFFICIENCY,FORE=>4,AFT=>2,EXP=>0);
NEW_LINE(3);

STOP;



VIDEO.CLEAR_SCREEN;

TEMPERATURE_EQUINOX:=(THERMAL_DISSIPATION
        /(RADIATOR_EMITTANCE_EOL*STEFAN_BOLTZMANN*EFFICIENCY
        *RADIATOR_HEIGHT*SOLAR_ARRAY_DIAMETER*PI))**0.25;
NEW_LINE(2);
PUT("Temperature at Equinox is ");
SET_COL(60);
PUT(TEMPERATURE_EQUINOX,FORE=>5,AFT=>2,EXP=>0);
PUT(" kelvin");

PUT(OUTH,"Temperature at Equinox is ");
SET_COL(OUTH,55);
PUT(OUTH,TEMPERATURE_EQUINOX,FORE=>5,AFT=>2,EXP=>0);
PUT(OUTH," kelvin");
NEW_LINE(OUTH,1);


TEMPERATURE_EQUINOX_CELCIUS:= TEMPERATURE_EQUINOX-273.15;
NEW_LINE(2);
PUT("Temperature at Equinox (celcius) is ");
SET_COL(60);
PUT(TEMPERATURE_EQUINOX_CELCIUS,FORE=>5,AFT=>2,EXP=>0);
PUT(" celcius");

PUT(OUTH,"Temperature at Equinox (celcius) is ");
SET_COL(OUTH,55);
PUT(OUTH,TEMPERATURE_EQUINOX_CELCIUS,FORE=>5,AFT=>2,EXP=>0);
PUT(OUTH," celcius");
NEW_LINE(OUTH,1);
```

```
if TEMPERATURE_EQUINOX_CELCIUS >= 5.0
   and TEMPERATURE_EQUINOX_CELCIUS <= 37.0  then
  NEW_LINE(2);
  PUT_LINE("Temperature during equinox non eclipse period is within ");
  PUT_LINE("the prescribed limits of 5 to 37 degrees celcius.  No ");
  PUT_LINE("auxiliary heating is required! ");
  PUT_LINE(OUTH,"Temperature during equinox non eclipse period is within ");
  PUT_LINE(OUTH,"the prescribed limits of 5 to 37 degrees celcius.  No ");
  PUT_LINE(OUTH,"auxiliary heating is required! ");
  NEW_LINE(OUTH,2);
elsif  TEMPERATURE_EQUINOX_CELCIUS < 5.0 then
  NEW_LINE(2);
  PUT_LINE(" Temperature is to low auxiliary heaters will be required");
  PUT_LINE(OUTH,"Temperature is to low, auxiliary heaters will be required");
  NEW_LINE(OUTH,2);
end if;
NEW_LINE(2);
PUT_LINE("****************************************************************");
NEW_LINE(3);


STOP;
VIDEO.CLEAR_SCREEN;

-- Equilibrium Temperature calculation

NEW_LINE(2);
PUT_LINE("When batteries provide partial power, heat dissipation  ");
PUT_LINE("and equilibrium temperature during eclipse is  ");
new_line(2);

EQUILIBRIUM_TEMPERATURE:=((THERMAL_DISSIPATION*PERCENT_PARTIAL_POWER)
                /(RADIATOR_EMITTANCE_EOL*STEFAN_BOLTZMANN
                *RADIATOR_HEIGHT*SOLAR_ARRAY_DIAMETER*PI))**0.25;
---------------------------------------------------------------------

SET_COL(45);
PUT(EQUILIBRIUM_TEMPERATURE,FORE=>5,AFT=>2,EXP=>0);
PUT("  degrees kelvin");

PUT(OUTH,"Equilibrium Temperature Partial Power Supplied");
SET_COL(OUTH,55);
PUT(OUTH,EQUILIBRIUM_TEMPERATURE,FORE=>5,AFT=>2,EXP=>0);
PUT(OUTH,"  kelvin");
NEW_LINE(OUTH,1);

new_line(2);
PUT_LINE("****************************************************************");
NEW_LINE(3);

-- Time Constant Calculation
```

```
   STOP;
   VIDEO.CLEAR_SCREEN;

   TIME_CONSTANT:=(MASS_RADIATOR_PLUS_EQUIPMENT*RADIATOR_SPECIFIC_HEAT)
                 /(4.0*RADIATOR_EMITTANCE_EOL*STEFAN_BOLTZMANN
                 *RADIATOR_HEIGHT*SOLAR_ARRAY_DIAMETER*PI
                 *EQUILIBRIUM_TEMPERATURE**3.0);
   NEW_LINE(2);
   PUT("Time Constant is ");
   SET_COL(55);
   PUT(TIME_CONSTANT,FORE=>5,AFT=>2,EXP=>0);
   PUT("  seconds");
   TIME_CONSTANT_MINUTES:=TIME_CONSTANT/60.0;
   NEW_LINE(2);
   PUT("Time Constant Minutes is ");
   SET_COL(50);
   PUT(TIME_CONSTANT_MINUTES,FORE=>5,AFT=>2,EXP=>0);
   PUT("  minutes");

   PUT(OUTH,"Time Constant is ");
   SET_COL(OUTH,53);
   PUT(OUTH,TIME_CONSTANT,FORE=>7,AFT=>0,EXP=>0);
   PUT(OUTH," seconds");
   NEW_LINE(OUTH,1);
   PUT(OUTH,"Time Constant Minutes is ");
   SET_COL(outh,55);
   PUT(OUTH,TIME_CONSTANT_MINUTES,FORE=>5,AFT=>2,EXP=>0);
   PUT(OUTH," minutes");
   NEW_LINE(OUTH,1);

   NEW_LINE(2);
   PUT_LINE("****************************************************************");
   NEW_LINE(2);
   STOP;
   VIDEO.CLEAR_SCREEN;


end if;

   PUT_LINE("For the case of radiative cooling find 'C' assuming t=0.0 ");
   PUT_LINE("this happens when the equinox temperature is less than the ");
   PUT_LINE("calculated equilibrium temperature");


   if (TEMPERATURE_EQUINOX-EQUILIBRIUM_TEMPERATURE ) <   5.0 then
     EQUILIBRIUM_TEMPERATURE:=EQUILIBRIUM_TEMPERATURE/1.0146;
     NEW_LINE(3);
     PUT_LINE("THE AMOUNT OF POWER PROVIDED DURING ECLIPSE IS TO CLOSE TO");
     PUT_LINE("FULL POWER SO A FUDGE FACTOR OF  1.0146% HAS BEEN SUBTRACTED");
     PUT_LINE("TO EQUILIBRIUM TO PREVENT A COTH NUMERIC ERROR.");
     NEW_LINE(3);
   end if;
```

```
 CONST:=2.0*(ARCCOTH(TEMPERATURE_EQUINOX/EQUILIBRIUM_TEMPERATURE)
          -ARCCOT(TEMPERATURE_EQUINOX/EQUILIBRIUM_TEMPERATURE));


 NEW_LINE(2);
 PUT("For radiative cooling constant 'C' when t=0.0 is ");
 SET_COL(60);
 PUT(CONST,FORE=>3,AFT=>6,EXP=>0);

 PUT(OUTH,"Radiative cooling constant 'C' when t=0.0 is ");
 SET_COL(OUTH,55);
 PUT(OUTH,CONST,FORE=>3,AFT=>4,EXP=>0);
 NEW_LINE(OUTH,1);

 NEW_LINE(2);
 PUT_LINE("The next portion of this program is an iterative approach to  ");
 PUT_LINE("find the temperature after equinox for an equinox period )");
 PUT_LINE("of 1.2 hours (72 minutes or 4,320 seconds)");
 NEW_LINE(1);
 PUT_LINE("The purpose of this iteration is to bracket an Eclipse Time ");
 PUT_LINE("of 72 minutes.  For example we want one value above (80 minutes) ");
 PUT_LINE("and one value below (68) minutes  a good starting temperature ");
 PUT_LINE("is 273.0 degrees kelvin. ");
 NEW_LINE(2);

 STOP;
 VIDEO.CLEAR_SCREEN;

<<EQUINOX_ONE>>



   PUT_LINE("Please enter an AFTER EQUINOX TEMPERATURE guess in degrees kelvin ");
   NEW_LINE(3);
   SET_COL(10);
   GET_DATA(EQUINOX_TEMP_ONE);
   VIDEO.CLEAR_SCREEN;
if EQUINOX_TEMP_ONE <= EQUILIBRIUM_TEMPERATURE then
   PUT_LINE ("Condition would not occur in real life please try again");
   NEW_LINE(1);
   PUT("Enter a value greater than ");
   PUT(EQUILIBRIUM_TEMPERATURE,FORE=>4,AFT=>2,EXP=>0);
   NEW_LINE(3);
   goto EQUINOX_ONE;
else
   TIME_ECLIPSE_ONE:=(2.0*(ARCCOTH(EQUINOX_TEMP_ONE/EQUILIBRIUM_TEMPERATURE)
             -ARCCOT(EQUINOX_TEMP_ONE/EQUILIBRIUM_TEMPERATURE))-CONST)
             *TIME_CONSTANT_MINUTES;
   PUT("Temperature Input is ");
   SET_COL(50);
```

```
    PUT(EQUINOX_TEMP_ONE,FORE=>5,AFT=>2,EXP=>0);
    PUT(" degrees kelvin");

    NEW_LINE(2);
    PUT("For radiative cooling Eclipse Time is ");
    SET_COL(60);
    PUT(TIME_ECLIPSE_ONE,FORE=>5,AFT=>2,EXP=>0);
    PUT(" minutes");


    new_line(2);
    PUT_LINE("Is the calculated eclipse time close to 72 minutes +/- 10 minutes ");
    PUT_LINE("Remember if this value is above 72 minutes then the next value ");
    PUT_LINE("should be below 72 minutes, or vice versus. ");
    PUT_LINE("To accept this eclipse time Enter 'Y' for yes  and  'N'  for  no ");
    if ABS(TIME_ECLIPSE_ONE-ECLIPSE_TIME_MINUTES) >10.0 then
        NEW_LINE(2);
        PUT_LINE("CALCULATIONS SHOW THAT ECLIPSE FOR TEMPERATURE IS MORE THAN 10
MINUTES");
        PUT_LINE("DIFFERENCE FROM THE 72 MINUTE STANDARD GEOSYNCHRONOUS ECLIPSE
TIME ");
        PUT_LINE("RECOMMEND INPUT A  'N' TO RE-CALCULATE A NEW ECLIPSE TIME");
        put_line("************************************************************************");

    end if;
    NEW_LINE(2);
    GET_CHARACTER(TAKE);
    if TAKE = 'Y' or TAKE = 'y' then
        TAKE := 'N' ;
        VIDEO.CLEAR_SCREEN;
        goto EQUINOX_TWO;
    else
        VIDEO.CLEAR_SCREEN;
        PUT_LINE(" Please enter a new temperature for the equinox temperature ");
        goto EQUINOX_ONE;
    end if;

    STOP;
    VIDEO.CLEAR_SCREEN;


    PUT_LINE("Please enter a new after equinox temperature guess in degrees kelvin");
end if;


<<EQUINOX_TWO>>
    NEW_LINE(2);
    PUT("First Iterative Eclipse Time for linear approximation is ");
    SET_COL(60);
    PUT(TIME_ECLIPSE_ONE,FORE=>5,AFT=>2,EXP=>0);
    PUT(" minutes");
    NEW_LINE(2);
```

```
      PUT("First Iterative Temperature is ");
      SET_COL(50);
      PUT(EQUINOX_TEMP_ONE,FORE=>5,AFT=>2,EXP=>0);
      PUT("  degrees kelvin");
      new_line(2);



      PUT_LINE("Please enter a new after equinox temperature guess in degrees kelvin");
      NEW_LINE(1);
      GET_DATA(EQUINOX_TEMP_TWO);
   if EQUINOX_TEMP_TWO <= EQUILIBRIUM_TEMPERATURE  then
          VIDEO.CLEAR_SCREEN;
          PUT_LINE ("Condition would not occur in real life please try again");
          NEW_LINE(1);
          PUT("Enter a value greater than ");
          PUT(EQUILIBRIUM_TEMPERATURE,FORE=>4,AFT=>2,EXP=>0);
          NEW_LINE(3);
          goto EQUINOX_TWO;
   else
      VIDEO.CLEAR_SCREEN;
      TIME_ECLIPSE_TWO:= (2.0*(ARCCOTH(EQUINOX_TEMP_TWO/EQUILIBRIUM_TEMPERATURE)
                          -ARCCOT(EQUINOX_TEMP_TWO/EQUILIBRIUM_TEMPERATURE))-CONST)
                          *TIME_CONSTANT_MINUTES;
      PUT("First Iterative Eclipse Time for linear approximation is ");
      SET_COL(60);
      PUT(TIME_ECLIPSE_ONE,FORE=>5,AFT=>2,EXP=>0);
      PUT("  minutes");
      NEW_LINE(2);
      PUT("First Iterative Temperature is ");
      SET_COL(50);
      PUT(EQUINOX_TEMP_ONE,FORE=>5,AFT=>2,EXP=>0);
      PUT("  degrees kelvin");
      NEW_LINE(2);
      PUT("Second Iterative Eclipse Time for linear approximation is ");
      SET_COL(60);
      PUT(TIME_ECLIPSE_TWO,FORE=>5,AFT=>2,EXP=>0);
      PUT("  minutes");
      new_line(2);
      PUT("Second Iterative Temperature is ");
      SET_COL(50);
      PUT(EQUINOX_TEMP_TWO,FORE=>5,AFT=>2,EXP=>0);
      PUT("  degrees kelvin");
      new_line(2);
      if ABS(TIME_ECLIPSE_TWO-ECLIPSE_TIME_MINUTES) > 10.0 then
          NEW_LINE(2);
          PUT_LINE("CALCULATIONS SHOW THAT ECLIPSE FOR TEMPERATURE IS GREATER THAN
10 MINUTES");
          PUT_LINE("RECOMMEND INPUT A  'N' TO RE-CALCULATED A NEW ECLIPSE TIME");
          put_line("*****************************************************************************");

      end if;
```

```
    PUT_LINE("Is the calculated eclipse time close to 72 minutes +/- 10 minutes");
    PUT_LINE("Remember if this value is above 72 minutes then the next value");
    PUT_LINE("should below 72 minutes");
    PUT_LINE("To accept this eclipse time Enter 'Y' for yes  and  'N'  for  no");
    SET_COL(15);
    GET_CHARACTER(TAKE);
    VIDEO.CLEAR_SCREEN;
    if TAKE = 'Y' or TAKE = 'y' then
        if TIME_ECLIPSE_ONE = 72.0  then
          TEMPERATURE_AFTER_EQUINOX := EQUINOX_TEMP_ONE;
          goto TAE;
        elsif TIME_ECLIPSE_TWO = 72.0  then
          TEMPERATURE_AFTER_EQUINOX := EQUINOX_TEMP_TWO;
          goto TAE;
        elsif TIME_ECLIPSE_ONE < 72.0  and  TIME_ECLIPSE_TWO > 72.0 then
          TAKE := 'N' ;
          goto STOP_ITERATION;
        elsif TIME_ECLIPSE_ONE > 72.0  and  TIME_ECLIPSE_TWO < 72.0 then
          TAKE := 'N' ;
          goto STOP_ITERATION;
        elsif TIME_ECLIPSE_ONE < 72.0  and  TIME_ECLIPSE_TWO < 72.0 then
          PUT_LINE("BOTH TIME VALUES ARE BELOW 72.0 MINUTES TRY AGAIN");
          TAKE := 'N' ;
          goto EQUINOX_TWO;
        elsif TIME_ECLIPSE_ONE > 72.0  and  TIME_ECLIPSE_TWO > 72.0 then
          PUT_LINE("BOTH TIME VALUES ARE ABOVE 72.0 MINUTES TRY AGAIN");
          TAKE := 'N' ;
          goto EQUINOX_TWO;
        end if;

    else
          VIDEO.CLEAR_SCREEN;
          PUT_LINE("Please enter a new temperature in order to bracket a 72 minute eclipse time");
          TAKE := 'N' ;
          goto EQUINOX_TWO;
    end if;
end if;
-- CASE OF RADIATIVE HEATING




<<STOP_ITERATION>>


  TEMPERATURE_AFTER_EQUINOX:=ABS(((TIME_ECLIPSE_ONE-ECLIPSE_TIME_MINUTES)
                 /(TIME_ECLIPSE_ONE-TIME_ECLIPSE_TWO))
                 *(EQUINOX_TEMP_ONE-EQUINOX_TEMP_TWO)
                 -EQUINOX_TEMP_ONE);
```

```
<<TAE>>

  NEW_LINE(OUTH,1);
  PUT(OUTH,"First Iterative Time for linear approximation is ");
  SET_COL(OUTH,55);
  PUT(OUTH,TIME_ECLIPSE_ONE,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH," minutes");
  NEW_LINE(OUTH,1);

  PUT(OUTH,"First Iterative Temperature is ");
  SET_COL(OUTH,55);
  PUT(OUTH,EQUINOX_TEMP_ONE,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH," degrees kelvin");
  new_line(OUTH,1);


  PUT(OUTH,"Second Iterative Time for linear approximation is ");
  SET_COL(OUTH,55);
  PUT(OUTH,TIME_ECLIPSE_TWO,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH," minutes");
  new_line(OUTH,1);
  PUT(OUTH,"Second Iterative Temperature is ");
  SET_COL(OUTH,55);
  PUT(OUTH,EQUINOX_TEMP_TWO,FORE=>5,AFT=>2,EXP=>0);

  new_line(OUTH,1);


  PUT ("Temperature after Equinox is ");
  SET_COL(50);
  PUT(TEMPERATURE_AFTER_EQUINOX,FORE=>5,AFT=>2,EXP=>0);

  PUT (OUTH,"Temperature after Equinox is ");
  SET_COL(OUTH,55);
  PUT(OUTH,TEMPERATURE_AFTER_EQUINOX,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH," kelvin");
  NEW_LINE(OUTH,2);

  NEW_LINE(2);
  PUT_LINE("*****************************************************************");
  NEW_LINE(2);
  STOP;

end HEAT;

procedure WARM_UP (THERMAL_DISSIPATION          : in out FLOAT;
                   RADIATOR_SPECIFIC_HEAT       : in out FLOAT;
                   RADIATOR_EMITTANCE_EOL       : in out FLOAT;
                   RADIATOR_AREA                : in out FLOAT;
                   MASS_RADIATOR_PLUS_EQUIPMENT : in out FLOAT;
                   EFFICIENCY                   : in out FLOAT;
                   TEMPERATURE_AFTER_EQUINOX    : in out FLOAT;
```

```
                    NUMBER_THERMAL_EMITTING_FACES: in out FLOAT) is

  OPERATING_TEMPERATURE  : FLOAT := 278.0;-- degrees kelvin
  TIME_CONSTANT_MINUTES,
  CONST,
  TIME_COOLING,
  TIME_HEATING           : FLOAT ;


  NEW_TEMPERATURE        : INTEGER ;


begin
  VIDEO.CLEAR_SCREEN;
  PUT_LINE("Now we will determine the time it takes for the spacecraft to");
  PUT_LINE("reach a specified operating temperature after eclipse.  Default");
  PUT_LINE("operating temperature is specified as 278.0 degrees Kelvin or");
  PUT_LINE("5 degrees celcius.");
  new_line(1);
  PUT_LINE(" DEFAULT VALUES FOR OPERATING PARAMETERS ARE");
  PUT("Radiator(s) Dissipate ");
  SET_COL(50);
  PUT(THERMAL_DISSIPATION,FORE=>5,AFT=>2,EXP=>0);
  PUT("  watts");
  NEW_LINE(1);

  PUT("EFFICIENCY");
  SET_COL(50);
  PUT(EFFICIENCY,FORE=>5,AFT=>2,EXP=>0);
  NEW_LINE(1);

  PUT("RADIATOR_EMITTANCE_EOL");
  SET_COL(50);
  PUT(RADIATOR_EMITTANCE_EOL,FORE=>5,AFT=>2,EXP=>0);
  NEW_LINE(1);

  PUT("MASS_RADIATOR_PLUS_EQUIPMENT");
  SET_COL(50);
  PUT(MASS_RADIATOR_PLUS_EQUIPMENT,FORE=>5,AFT=>2,EXP=>0);
  PUT("  kgs");
  NEW_LINE(1);

  PUT("RADIATOR_SPECIFIC_HEAT");
  SET_COL(50);
  PUT(RADIATOR_SPECIFIC_HEAT,FORE=>5,AFT=>2,EXP=>0);
  PUT("(watts*sec)\(kg*Kelvin) ");
  NEW_LINE(1);

  PUT("Radiator Area is ");
  SET_COL(50);
  PUT(RADIATOR_AREA,FORE=>5,AFT=>2,EXP=>0);
  PUT("  meters^2");
```

```
NEW_LINE(1);

PUT("Temperature after equinox is ");
SET_COL(50);
PUT(TEMPERATURE_AFTER_EQUINOX,FORE=>5,AFT=>2,EXP=>0);
PUT(" kelvin");
NEW_LINE(2);


PUT("Desired operating temperature after equinox is ");
SET_COL(50);
PUT(OPERATING_TEMPERATURE,FORE=>5,AFT=>2,EXP=>0);
PUT(" kelvin");
NEW_LINE(1);

PUT_LINE("The Radiator Heat Dissipation may be changed if desired");
PUT_LINE("this value enter a 'y' for YES. If you wish to change the value");
PUT_LINE("enter a 'n' for NO and the value you enter will be used in ");
put_line("further calculations");
GET_CHARACTER(CHAR);
if CHAR = 'Y' or CHAR = 'y' then
  VIDEO.CLEAR_SCREEN;
  PUT("Please enter a value for the Radiator Heat Dissipation ");
  NEW_LINE(3);
  PUT_LINE("***************************************************************");
  NEW_LINE(3);
  GET_DATA(THERMAL_DISSIPATION);
  PUT("Radiator Heat Dissipation is  ");
  SET_COL(60);
  PUT(THERMAL_DISSIPATION, FORE => 6, AFT => 2, EXP => 0);
  NEW_LINE(1);
else
  VIDEO.CLEAR_SCREEN;
end if;
NEW_LINE(1);

PUT_LINE("The default operating temperature after equinox is 5 degrees celcius");
PUT_LINE("or 278 degrees kelvin.  If you wish to change this value enter a");
PUT_LINE("'y' for YES.  To accept the default value enter an 'n' for NO changes");
NEW_LINE(2);

GET_CHARACTER(CHAR);
if CHAR = 'Y' or CHAR = 'y' then
  VIDEO.CLEAR_SCREEN;
  PUT("Please enter a value for operating temperature after equinox ");
  NEW_LINE(3);
  PUT_LINE("***************************************************************");
  NEW_LINE(3);
  GET_DATA(OPERATING_TEMPERATURE);
  PUT("Desired operating temperature is  ");
  SET_COL(60);
  PUT(OPERATING_TEMPERATURE, FORE => 6, AFT => 2, EXP => 0);
  NEW_LINE(1);
else
  VIDEO.CLEAR_SCREEN;
end if;

-- Equilibrium Temperature calculation

PUT(OUTH,"Radiator(s) Dissipate ");
SET_COL(OUTH,55);
PUT(OUTH,THERMAL_DISSIPATION,FORE=>5,AFT=>2,EXP=>0);
PUT(OUTH," watts");
NEW_LINE(OUTH,1);

PUT(OUTH,"Efficiency");
SET_COL(OUTH,55);
PUT(OUTH,EFFICIENCY,FORE=>5,AFT=>2,EXP=>0);
NEW_LINE(OUTH,1);

PUT(OUTH,"Radiator Emittance");
SET_COL(OUTH,55);
PUT(OUTH,RADIATOR_EMITTANCE_EOL,FORE=>5,AFT=>2,EXP=>0);
NEW_LINE(OUTH,1);

PUT(OUTH,"Mass Radiator Plus Equipment");
SET_COL(OUTH,55);
PUT(OUTH,MASS_RADIATOR_PLUS_EQUIPMENT,FORE=>5,AFT=>2,EXP=>0);
PUT(OUTH," kgs");
NEW_LINE(OUTH,1);

PUT(OUTH,"Radiator Specific Heat");
SET_COL(OUTH,55);
PUT(OUTH,RADIATOR_SPECIFIC_HEAT,FORE=>5,AFT=>2,EXP=>0);
PUT(OUTH,"(W*s)/(kg*Kelvin) ");
NEW_LINE(OUTH,1);

PUT(OUTH,"Radiator Area is ");
SET_COL(OUTH,55);
PUT(OUTH,RADIATOR_AREA,FORE=>5,AFT=>2,EXP=>0);
PUT(OUTH," meters^2");
NEW_LINE(OUTH,1);

PUT(OUTH,"Desired operating temperature after equinox is ");
SET_COL(OUTH,55);
PUT(OUTH,OPERATING_TEMPERATURE,FORE=>5,AFT=>2,EXP=>0);
PUT(OUTH," kelvin");
NEW_LINE(OUTH,1);

EQUILIBRIUM_TEMPERATURE:=(THERMAL_DISSIPATION/
  (RADIATOR_EMITTANCE_EOL*STEFAN_BOLTZMANN*EFFICIENCY*RADIATOR_AREA))**0.25;
NEW_LINE(2);
PUT("Equilibrium Temperature");
SET_COL(50);
```

```
PUT(EQUILIBRIUM_TEMPERATURE,FORE=>7,AFT=>2,EXP=>0);
PUT(" kelvin");
STOP;

if (EQUILIBRIUM_TEMPERATURE-TEMPERATURE_AFTER_EQUINOX) <  5.0 then
  EQUILIBRIUM_TEMPERATURE:=1.0146*EQUILIBRIUM_TEMPERATURE;
  NEW_LINE(3);
  PUT_LINE("THE AMOUNT OF POWER PROVIDED DURING ECLIPSE IS TO CLOSE TO");
  PUT_LINE("FULL POWER SO A FUDGE FACTOR OF  1.0146% HAS BEEN ADDED TO");
  PUT_LINE("EQUILIBRIUM TO PREVENT A TANH NUMERIC ERROR.");
end if;

  PUT("Radiative Heating 'C' assuming t=0.0 is ");
  CONST:=2.0*((ARCTANH(TEMPERATURE_AFTER_EQUINOX/EQUILIBRIUM_TEMPERATURE))-
     (ARCTAN(TEMPERATURE_AFTER_EQUINOX/EQUILIBRIUM_TEMPERATURE)));
  SET_COL(48);
  PUT(CONST,FORE=>7,AFT=>4,EXP=>0);
  NEW_LINE(1);

PUT(OUTH,"Equilibrium Temperature Heating After Eclipse");
SET_COL(OUTH,55);
PUT(OUTH,EQUILIBRIUM_TEMPERATURE,FORE=>5,AFT=>2,EXP=>0);
PUT(OUTH," kelvin");
new_line(OUTH,1);



-- Time Constant Calculation

TIME_CONSTANT:=(MASS_RADIATOR_PLUS_EQUIPMENT*RADIATOR_SPECIFIC_HEAT)
            /(4.0*RADIATOR_EMITTANCE_EOL*EFFICIENCY*STEFAN_BOLTZMANN
            *RADIATOR_AREA*EQUILIBRIUM_TEMPERATURE**3.0);
NEW_LINE(2);
PUT("Time Constant Heating After Eclipse is ");
SET_COL(50);
PUT(TIME_CONSTANT,FORE=>7,AFT=>2,EXP=>0);
PUT(" seconds");
NEW_LINE(2);

PUT(OUTH,"Time Constant Heating After Eclipse is");
SET_COL(OUTH,54);
PUT(OUTH,TIME_CONSTANT,FORE=>6,AFT=>2,EXP=>0);
PUT(OUTH," seconds");
NEW_LINE(OUTH,1);

TIME_CONSTANT_MINUTES:=TIME_CONSTANT/60.0;
PUT("Time Constant is ");
SET_COL(50);
PUT(TIME_CONSTANT_MINUTES,FORE=>7,AFT=>2,EXP=>0);
PUT(" minutes");
NEW_LINE(2);
```

```
PUT(OUTH,"Time Constant Heating After Eclipse is");
SET_COL(OUTH,55);
PUT(OUTH,TIME_CONSTANT_MINUTES,FORE=>5,AFT=>2,EXP=>0);
PUT(OUTH," minutes");
NEW_LINE(OUTH,1);


PUT("Temperature After Equinox");
SET_COL(50);
PUT(TEMPERATURE_AFTER_EQUINOX,FORE=>7,AFT=>2,EXP=>0);
PUT(" kelvin");
NEW_LINE(2);

PUT("Equilibrium Temperature");
SET_COL(50);
PUT(EQUILIBRIUM_TEMPERATURE,FORE=>7,AFT=>2,EXP=>0);
PUT(" kelvin");
NEW_LINE(2);

NEW_LINE(OUTH,1);
PUT(OUTH,"Radiative Cooling Constant when t=0.0");
SET_COL(OUTH,55);
PUT(OUTH,CONST,FORE=>3,AFT=>4,EXP=>0);
NEW_LINE(OUTH,1);

NEW_LINE(1);
PUT("Operating Temperature");
SET_COL(50);
PUT(OPERATING_TEMPERATURE,FORE=>7,AFT=>2,EXP=>0);
PUT(" kelvin");
new_LINE(2);

PUT(OUTH,"Operating Temperature Satellite");
SET_COL(OUTH,55);
PUT(OUTH,OPERATIN[illegible]_TEMPERATURE,FORE=>5,AFT=>2,EXP=>0);
PUT(OUTH," kelvin");
NEW_LINE(OUTH,1);

TIME_HEATING:= ((2.0*(ARCTANH(OPERATING_TEMPERATURE/EQUILIBRIUM_TEMPERATURE)
-ARCTAN(OPERATING_TEMPERATURE/EQUILIBRIUM_TEMPERATURE)))-CONST)
*TIME_CONSTANT_MINUTES;

NEW_LINE(2);
PUT("Radiative heating after Eclipse ");
SET_COL(50);
PUT(TIME_HEATING,FORE=>7,AFT=>2,EXP=>0);
PUT(" minutes");
NEW_LINE(2);

PUT(OUTH,"Time Constant");
SET_COL(OUTH,55);
PUT(OUTH,TIME_CONSTANT_MINUTES,FORE=>5,AFT=>2,EXP=>0);
```

```
  PUT(OUTH," minutes");
  NEW_LINE(OUTH,1);



  PUT(OUTH,"Time Radiative heating after Eclipse");
  SET_COL(OUTH,55);
  PUT(OUTH,TIME_HEATING,FORE=>5,AFT=>2,EXP=>0);
  PUT(OUTH," minutes");
  NEW_LINE(OUTH,1);

  STOP;


end WARM_UP;



begin


    CREATE(OUTH,NAME=>"THERMAL.DAT");

    PRINT_HEADER;

    DUAL_SPIN              (DRUM_SPINNER);

    OPERATING_DATA            (BATTERY_LOAD,
                              DRUM_SPINNER,
                              RADIATOR_SPECIFIC_HEAT,
                              RADIATOR_EMITTANCE_EOL,
                              MASS_RADIATOR_PLUS_EQUIPMENT,
                              EFFICIENCY,
                              NUMBER_THERMAL_EMITTING_FACES,
                              PAYLOAD_POWER,
                              ECLIPSE_TIME);

    HEAT                  (BATTERY_LOAD,
                              DRUM_SPINNER,
                              RADIATOR_AREA,
                              PAYLOAD_POWER,
                              TEMPERATURE_AFTER_EQUINOX,
                              THERMAL_DISSIPATION,
                              ECLIPSE_TIME);

    WARM_UP                (THERMAL_DISSIPATION,
                              RADIATOR_SPECIFIC_HEAT,
                              RADIATOR_EMITTANCE_EOL,
                              RADIATOR_AREA,
                              MASS_RADIATOR_PLUS_EQUIPMENT,
                              EFFICIENCY,
                                        TEMPERATURE_AFTER_EQUINOX,
                                        NUMBER_THERMAL_EMITTING_FACES);


    CLOSE(OUTH);

    STOP;
    NEW_LINE(2);
    PUT_LINE("DATA FOR THIS DESIGN RUN IS LOCATED IN THE FOLLOWING FILE");
    NEW_LINE(1);
    PUT_LINE(">>>>>>>>>>>>>>>>>>>>  THERMAL.DAT ");
    NEW_LINE(2);
    PUT_LINE("TO KEEP DATA FROM BEING ERASED ON NEXT RUN");
    PUT_LINE("USE DOS COMMAND REN (RENAME) ");
    NEW_LINE(1);
    PUT_LINE("EXAMPLE - REN THERMAL.DAT   THERMAL.INI");
    PUT_LINE("The .INI could be your initials");
    NEW_LINE(5);

end THERMAL;
```

D. ARRAY THERMAL CONTROL

```
-- Title          : Thermal Characteristics
-- Author         : David Lashbrook
-- Date           : 15 February 1992
-- Revised        : 30 March 1992
-- Compiler       : OPENADA EXT
-- Description    : This procedure determines the thermal characteristics for
--                  solar arrays in geosynchronous orbits.

with TEXT_IO, GETDATA, GENERIC_ELEMENTARY_FUNCTIONS, VIDEO;
use  TEXT_IO , GETDATA ;

procedure ARRAY_THERMAL_CONTROL is


  package FLOAT_INOUT is new FLOAT_IO(FLOAT);
  use     FLOAT_INOUT;
  package INTEGER_INOUT is new INTEGER_IO(INTEGER);
  use     INTEGER_INOUT;
  package BOOLEAN_INOUT is new ENUMERATION_IO(BOOLEAN);
  use     BOOLEAN_INOUT;
  package GEF_INOUT is new GENERIC_ELEMENTARY_FUNCTIONS(FLOAT);
  use     GEF_INOUT;


  OKAY                    : BOOLEAN := TRUE;
  DRUM_SPINNER                : BOOLEAN := FALSE;

  Y,
  y,
  N,
  n,
  TAKE,
  CHAR                 : CHARACTER ;


  DECISION,
  J                  : INTEGER ;

  PI                 : FLOAT := 3.14159;

  X                  : FLOAT ;

  OUTATC               :FILE_TYPE;




 procedure PRINT_HEADER is
```

```
  begin
    VIDEO.CLEAR_SCREEN;SET_LINE(1);
    NEW_LINE(2);
    SET_COL(10);
    PUT_LINE("This program walks through a ARRAY THERMAL Characteristics ");
    SET_COL(10);
    PUT_LINE("for a solar powered geosynchronous satellite.");
    SET_COL(10);
    PUT_LINE("All pertinent data will be saved to a file called ARRAYTC.DAT");
    NEW_LINE;
  end PRINT_HEADER;

procedure DUAL_SPIN (DRUM_SPINNER : in out BOOLEAN) is
begin
  SET_COL(10);
  PUT_LINE("Is your spacecraft Spin Stabilized ");
  SET_COL(15);
  GET_CHARACTER(char);
  if CHAR = 'Y' or CHAR = 'y' then
    DRUM_SPINNER:=TRUE;
    if DRUM_SPINNER = TRUE then
        VIDEO.CLEAR_SCREEN;SET_LINE(1);
        SET_COL(10);
        PUT_LINE("Satellite is Spin Stabilized");
        NEW_LINE(OUTATC,1);
        PUT_LINE(OUTATC,"Satellite is Spin Stabilized");
        PUT_LINE(OUTATC,"****************************");
        NEW_LINE(OUTATC,1);
        NEW_LINE(2);
        PUT_LINE("*****************************************************************");
    end if;
  else
    VIDEO.CLEAR_SCREEN;SET_LINE(1);
    SET_COL(10);
    PUT_LINE("Satellite is Three Axis Stabilized");
    NEW_LINE(OUTATC,1);
    PUT_LINE(OUTATC,"Satellite is Three Axis Stabilized");
    PUT_LINE(OUTATC,"**********************************");
    NEW_LINE(OUTATC,1);
    NEW_LINE(2);
    PUT_LINE("*****************************************************************");
  end if;
end DUAL_SPIN;



procedure SOLAR_ARRAY_TEMPERATURE (DRUM_SPINNER  : in out BOOLEAN) is

  STEFAN_BOLTZMANN                    : FLOAT := 5.67E-08;
  SOLAR_ASPECT_COEFFICIENT_SOLSTICE        : FLOAT := 23.5;
  SOLAR_ASPECT_COEFFICIENT_EQUINOX       : FLOAT := 0.0;
  CELL_EMITTANCE_FRONT                   : FLOAT := 0.8;
  CELL_EMITTANCE_BACK                    : FLOAT := 0.7;
  SOLAR_INTENSITY_WINTER_SOLSTICE        : FLOAT := 1397.0;  -- W/m^2
  SOLAR_INTENSITY_SUMMER_SOLSTICE        : FLOAT := 1311.0;  -- W/m^2
  SOLAR_INTENSITY_VERNAL_EQUINOX         : FLOAT := 1362.0;  -- W/m^2
  SOLAR_INTENSITY_AUTUMNAL_EQUINOX       : FLOAT := 1345.0;  -- W/m^2
  CELL_EFFICIENCY                        : FLOAT := 0.14;
  PACKING_FACTOR                         : FLOAT := 0.95;
  AVG_SOLAR_CELL_ABSORBTANCE             : FLOAT := 0.8;

  WINTER_SOLSTICE_OPERATING_TEMPERATURE,
  SUMMER_SOLSTICE_OPERATING_TEMPERATURE,
  VERNAL_EQUINOX_OPERATING_TEMPERATURE,
  AUTUMNAL_EQUINOX_OPERATING_TEMPERATURE,
  EFFECTIVE_SOLAR_CELL_ABSORBTANCE,
  FRONT_ARRAY_AREA,
  BACK_ARRAY_AREA                        : FLOAT ;



  CHANGE_SOLAR                           : INTEGER ;

begin
  PUT_LINE("This portion of the design uses the following values as listed below:");
  NEW_LINE(1);
  PUT_LINE("______________________________________________________________");
  NEW_LINE(1);
  PUT("Solstice Angle");
  SET_COL(55);
  PUT(SOLAR_ASPECT_COEFFICIENT_SOLSTICE,FORE=>4,AFT=>2,EXP=>0);
  PUT(" degrees");

  NEW_LINE(1);
  PUT("Equinox Angle");
  SET_COL(55);
  PUT(SOLAR_ASPECT_COEFFICIENT_EQUINOX,FORE=>4,AFT=>2,EXP=>0);
  PUT(" degrees");

  NEW_LINE(1);
  PUT("Cell Emittance Front");
  SET_COL(55);
  PUT(CELL_EMITTANCE_FRONT,FORE=>3,AFT=>3,EXP=>0);

  NEW_LINE(1);
  PUT("Cell Emittance Back");
  SET_COL(55);
  PUT(CELL_EMITTANCE_BACK,FORE=>3,AFT=>3,EXP=>0);

  NEW_LINE(1);
  PUT("Solar Intensity Winter Solstice");
  SET_COL(55);
  PUT(SOLAR_INTENSITY_WINTER_SOLSTICE,FORE=>4,AFT=>2,EXP=>0);
  PUT(" W/m^2");

  NEW_LINE(1);
  PUT("Solar Intensity Summer Solstice");
  SET_COL(55);
  PUT(SOLAR_INTENSITY_SUMMER_SOLSTICE,FORE=>4,AFT=>2,EXP=>0);
  PUT(" W/m^2");

  NEW_LINE(1);
  PUT("Solar Intensity Vernal Equinox");
  SET_COL(55);
  PUT(SOLAR_INTENSITY_VERNAL_EQUINOX,FORE=>4,AFT=>2,EXP=>0);
  PUT(" W/m^2");

  NEW_LINE(1);
  PUT("Solar Intensity Autumnal Equinox");
  SET_COL(55);
  PUT(SOLAR_INTENSITY_AUTUMNAL_EQUINOX,FORE=>4,AFT=>2,EXP=>0);
  PUT(" W/m^2");

  NEW_LINE(1);
  PUT("Cell Efficiency");
  SET_COL(55);
  PUT(CELL_EFFICIENCY,FORE=>2,AFT=>4,EXP=>0);

  NEW_LINE(1);
  PUT("Packing Factor");
  SET_COL(55);
  PUT(PACKING_FACTOR,FORE=>2,AFT=>4,EXP=>0);

  NEW_LINE(1);
  PUT("Average Solar Cell Absorbtance");
  SET_COL(55);
  PUT(AVG_SOLAR_CELL_ABSORBTANCE,FORE=>2,AFT=>4,EXP=>0);
  NEW_LINE(2);

  PUT_LINE("If you desire to CHANGE any of the listed values please enter ");
  PUT_LINE("a 'y' for YES  otherwise enter a 'n' for NO");
  GET_CHARACTER(CHAR);
  if CHAR = 'Y' or CHAR = 'y' then
    CHAR := N;
  VIDEO.CLEAR_SCREEN;

<<VALUE>>

  PUT_LINE("Please enter the number value to the right of the value you ");
  PUT_LINE("wish to CHANGE.        (DEFAULT VALUES ARE IN PARENTHESIS)");
  new_line;
  PUT_LINE("Cell emittance Back is not used for spin stabilized spacecraft calculations");
  put_LINE("------------------------------------------------------------------------");
  PUT_LINE("SOLAR_ASPECT_COEFFICIENT_SOLSTICE          [1]");
PUT_LINE("SOLAR_ASPECT_COEFFICIENT_EQUINOX          [2]");
PUT_LINE("CELL_EMITTANCE_FRONT                    [3]");
PUT_LINE("CELL_EMITTANCE_BACK                    [4]");
PUT_LINE("SOLAR_INTENSITY_WINTER_SOLSTICE          [5]");
PUT_LINE("SOLAR_INTENSITY_SUMMER_SOLSTICE          [6]");
PUT_LINE("SOLAR_INTENSITY_VERNAL_EQUINOX           [7]");
PUT_LINE("SOLAR_INTENSITY_AUTUMNAL_EQUINOX          [8]");
PUT_LINE("CELL_EFFICIENCY                       [9]");
PUT_LINE("PACKING_FACTOR                        [10]");
PUT_LINE("AVG_SOLAR_CELL_ABSORBTANCE               [11]");
GET_INTEGER(CHANGE_SOLAR);
VIDEO.CLEAR_SCREEN;

case CHANGE_SOLAR is
  when 1 =>
      VIDEO.CLEAR_SCREEN;
      PUT("Please enter a value for SOLAR_ASPECT_COEFFICIENT_SOLSTICE");
      NEW_LINE(3);
      GET_DATA(SOLAR_ASPECT_COEFFICIENT_SOLSTICE);
      NEW_LINE(1);
      PUT("Solstice Angle");
      SET_COL(60);
      PUT(SOLAR_ASPECT_COEFFICIENT_SOLSTICE,FORE=>4,AFT=>2,EXP=>0);
      PUT("  degrees");
      NEW_LINE(3);

  when 2 =>
      VIDEO.CLEAR_SCREEN;
      PUT("Please enter a value for SOLAR_ASPECT_COEFFICIENT_EQUINOX");
      NEW_LINE(3);
      GET_DATA(SOLAR_ASPECT_COEFFICIENT_EQUINOX);
      NEW_LINE(1);
      PUT("Equinox Angle");
      SET_COL(55);
      PUT(SOLAR_ASPECT_COEFFICIENT_EQUINOX,FORE=>4,AFT=>2,EXP=>0);
      PUT("  degrees");
      NEW_LINE(3);

  when 3 =>
      VIDEO.CLEAR_SCREEN;
      PUT("Please enter a value for CELL_EMITTANCE_FRONT");
      NEW_LINE(3);
      GET_DATA(CELL_EMITTANCE_FRONT);
      NEW_LINE(1);
      PUT("Cell Emittance Front ");
      SET_COL(55);
      PUT(CELL_EMITTANCE_FRONT,FORE=>4,AFT=>2,EXP=>0);
      NEW_LINE(3);
  when 4 =>
      VIDEO.CLEAR_SCREEN;
      NEW_LINE(3);
      PUT("Please enter a value for CELL_EMITTANCE_BACK");
```

```
    GET_DATA(CELL_EMITTANCE_BACK);
    NEW_LINE(1);
    PUT("Cell Emittance Back ");
    SET_COL(55);
    PUT(CELL_EMITTANCE_BACK,FORE=>4,AFT=>2,EXP=>0);
    NEW_LINE(3);
when 5 =>
    VIDEO.CLEAR_SCREEN;
    PUT("Please enter a value for SOLAR_INTENSITY_WINTER_SOLSTICE");
    NEW_LINE(3);
    GET_DATA(SOLAR_INTENSITY_WINTER_SOLSTICE);
    NEW_LINE(1);
    PUT("Solar Intensity Winter Solstice");
    SET_COL(55);
    PUT(SOLAR_INTENSITY_WINTER_SOLSTICE,FORE=>4,AFT=>2,EXP=>0);
    PUT("  W/m^2");
    NEW_LINE(3);
when 6 =>
    VIDEO.CLEAR_SCREEN;
    PUT("Please enter a value for SOLAR_INTENSITY_SUMMER_SOLSTICE");
    NEW_LINE(3);
    GET_DATA(SOLAR_INTENSITY_SUMMER_SOLSTICE);
    NEW_LINE(1);
    PUT("Solar Intensity Summer Solstice");
    SET_COL(55);
    PUT(SOLAR_INTENSITY_SUMMER_SOLSTICE,FORE=>4,AFT=>2,EXP=>0);
    PUT("  W/m^2");
    NEW_LINE(3);

when 7 =>
    VIDEO.CLEAR_SCREEN;
    PUT("Please enter a value for SOLAR_INTENSITY_VERNAL_EQUINOX");
    NEW_LINE(3);
    GET_DATA(SOLAR_INTENSITY_VERNAL_EQUINOX);
    NEW_LINE(1);
    PUT("Solar Intensity Vernal Equinox");
    SET_COL(55);
    PUT(SOLAR_INTENSITY_VERNAL_EQUINOX,FORE=>4,AFT=>2,EXP=>0);
    PUT("  W/m^2");
    NEW_LINE(3);
when 8 =>
    VIDEO.CLEAR_SCREEN;
    PUT("Please enter a value for SOLAR_INTENSITY_AUTUMNAL_EQUINOX");
    NEW_LINE(3);
    GET_DATA(SOLAR_INTENSITY_AUTUMNAL_EQUINOX);
    NEW_LINE(1);
    PUT("Solar Intensity Autumnal Equinox");
    SET_COL(55);
    PUT(SOLAR_INTENSITY_AUTUMNAL_EQUINOX,FORE=>4,AFT=>2,EXP=>0);
    PUT("  W/m^2");
    NEW_LINE(3);
when 9 =>
```

```
        VIDEO.CLEAR_SCREEN;
        PUT("Please enter a value for CELL_EFFICIENCY");
        NEW_LINE(3);
        GET_DATA(CELL_EFFICIENCY);
        NEW_LINE(1);
        PUT("Cell Efficiency");
        SET_COL(55);
        PUT(CELL_EFFICIENCY,FORE=>2,AFT=>4,EXP=>0);
        NEW_LINE(3);

    when 10 =>
        VIDEO.CLEAR_SCREEN;
        PUT("Please enter a value for PACKING_FACTOR");
        NEW_LINE(3);
        GET_DATA(PACKING_FACTOR);
        NEW_LINE(1);
        PUT("Packing Factor");
        SET_COL(55);
        PUT(PACKING_FACTOR,FORE=>2,AFT=>4,EXP=>0);
        NEW_LINE(3);

    when 11 =>
        VIDEO.CLEAR_SCREEN;
        PUT("Please enter a value for AVG_SOLAR_CELL_ABSORBTANCE");
        NEW_LINE(3);
        GET_DATA(AVG_SOLAR_CELL_ABSORBTANCE);
        NEW_LINE(1);
        PUT("Average Solar Cell Absorbtance");
        SET_COL(55);
        PUT(AVG_SOLAR_CELL_ABSORBTANCE,FORE=>2,AFT=>4,EXP=>0);
        NEW_LINE(3);

    when others =>
        VIDEO.CLEAR_SCREEN;
        PUT_LINE("Thank You for your input ");
        NEW_LINE(4);
  end case;

  CHAR := N;
        NEW_LINE(4);
        PUT_LINE("If you wish to change another value please enter a 'y' for YES");
        PUT_LINE("otherwise enter a 'n' for NO ");
        GET_CHARACTER(CHAR);
        if CHAR = 'Y' or CHAR = 'y' then
             CHAR := N;
             VIDEO.CLEAR_SCREEN;
             goto VALUE;
        else
          VIDEO.CLEAR_SCREEN;
          PUT_LINE("UNDERSTAND NO MORE CHANGES");
          NEW_LINE(3);
        end if;
  else
      VIDEO.CLEAR_SCREEN;
      PUT_LINE("UNDERSTAND DEFAULT VALUES WILL BE USED");
      NEW_LINE(3);
  end if;


NEW_LINE(OUTATC,1);
PUT(OUTATC,"Solstice Angle");
SET_COL(OUTATC,55);
PUT(OUTATC,SOLAR_ASPECT_COEFFICIENT_SOLSTICE,FORE=>4,AFT=>2,EXP=>0);
PUT(OUTATC," degrees");

NEW_LINE(OUTATC,2);
PUT(OUTATC,"Equinox Angle");
SET_COL(OUTATC,55);
PUT(OUTATC,SOLAR_ASPECT_COEFFICIENT_EQUINOX,FORE=>4,AFT=>2,EXP=>0);
PUT(OUTATC," degrees");

NEW_LINE(OUTATC,2);
PUT(OUTATC,"Cell Emittance Front");
SET_COL(OUTATC,55);
PUT(OUTATC,CELL_EMITTANCE_FRONT,FORE=>3,AFT=>3,EXP=>0);

NEW_LINE(OUTATC,2);
PUT(OUTATC,"Cell Emittance Back");
SET_COL(OUTATC,55);
PUT(OUTATC,CELL_EMITTANCE_BACK,FORE=>3,AFT=>3,EXP=>0);

NEW_LINE(OUTATC,2);
PUT(OUTATC,"Solar Intensity Winter Solstice");
SET_COL(OUTATC,55);
PUT(OUTATC,SOLAR_INTENSITY_WINTER_SOLSTICE,FORE=>4,AFT=>2,EXP=>0);
PUT(OUTATC," W/m^2");

NEW_LINE(OUTATC,2);
PUT(OUTATC,"Solar Intensity Summer Solstice");
SET_COL(OUTATC,55);
PUT(OUTATC,SOLAR_INTENSITY_SUMMER_SOLSTICE,FORE=>4,AFT=>2,EXP=>0);
PUT(OUTATC," W/m^2");

NEW_LINE(OUTATC,2);
PUT(OUTATC,"Solar Intensity Vernal Equinox");
SET_COL(OUTATC,55);
PUT(OUTATC,SOLAR_INTENSITY_VERNAL_EQUINOX,FORE=>4,AFT=>2,EXP=>0);
PUT(OUTATC," W/m^2");

NEW_LINE(OUTATC,2);
PUT(OUTATC,"Solar Intensity Autumnal Equinox");
SET_COL(OUTATC,55);
PUT(OUTATC,SOLAR_INTENSITY_AUTUMNAL_EQUINOX,FORE=>4,AFT=>2,EXP=>0);
PUT(OUTATC," W/m^2");
```

```
  NEW_LINE(OUTATC,2);
  PUT(OUTATC,"Cell Efficiency");
  SET_COL(OUTATC,55);
  PUT(OUTATC,CELL_EFFICIENCY,FORE=>2,AFT=>4,EXP=>0);

  NEW_LINE(OUTATC,2);
  PUT(OUTATC,"Packing Factor");
  SET_COL(OUTATC,55);
  PUT(OUTATC,PACKING_FACTOR,FORE=>2,AFT=>4,EXP=>0);

  NEW_LINE(OUTATC,2);
  PUT(OUTATC,"Average Solar Cell Absorbtance");
  SET_COL(OUTATC,55);
  PUT(OUTATC,AVG_SOLAR_CELL_ABSORBTANCE,FORE=>2,AFT=>4,EXP=>0);

  VIDEO.CLEAR_SCREEN;

  EFFECTIVE_SOLAR_CELL_ABSORBTANCE:= AVG_SOLAR_CELL_ABSORBTANCE
                                     -PACKING_FACTOR*CELL_EFFICIENCY;
  NEW_LINE(2);
  PUT("Effective Solar Cell Absorbtance is ");
  NEW_LINE(3);
  SET_COL(55);
  PUT(EFFECTIVE_SOLAR_CELL_ABSORBTANCE,FORE=>2,AFT=>4,EXP=>0);


  NEW_LINE(OUTATC,2);
  PUT(OUTATC,"Effective Solar Cell Absorbtance is ");
  SET_COL(OUTATC,55);
  PUT(OUTATC,EFFECTIVE_SOLAR_CELL_ABSORBTANCE,FORE=>2,AFT=>4,EXP=>0);



if DRUM_SPINNER = FALSE then

  NEW_LINE(4);
  PUT("Please enter the FRONT solar cell array area in meters squared ");
  NEW_LINE(3);
  GET_DATA(FRONT_ARRAY_AREA);
  NEW_LINE(3);
  PUT("Front Array Area is ");
  set_col(55);
  PUT(FRONT_ARRAY_AREA,FORE=>3,AFT=>3,EXP=>0);
  PUT("    m^2");

  NEW_LINE(4);
  PUT("Please enter the BACK solar cell array area in meters squared ");
  NEW_LINE(3);
  GET_DATA(BACK_ARRAY_AREA);
  VIDEO.CLEAR_SCREEN;
  NEW_LINE(3);
  PUT("Front Array Area is ");
```

```
set_col(55);
PUT(FRONT_ARRAY_AREA,FORE=>3,AFT=>3,EXP=>0);
PUT("    m^2");
NEW_LINE(2);
PUT("Back Array Area is ");
set_col(55);
PUT(BACK_ARRAY_AREA,FORE=>3,AFT=>3,EXP=>0);
PUT("    m^2");
NEW_LINE(5);
STOP;


WINTER_SOLSTICE_OPERATING_TEMPERATURE:=((EFFECTIVE_SOLAR_CELL_ABSORBTANCE
   *FRONT_ARRAY_AREA
   *SOLAR_INTENSITY_WINTER_SOLSTICE
   *COS(SOLAR_ASPECT_COEFFICIENT_SOLSTICE*PI/180.0))

   /((CELL_EMITTANCE_FRONT*FRONT_ARRAY_AREA+CELL_EMITTANCE_BACK
   *BACK_ARRAY_AREA)*STEFAN_BOLTZMANN))**0.25;

NEW_LINE(2);
PUT("Winter Solstice Operating Temperature is ");
SET_COL(55);
PUT(WINTER_SOLSTICE_OPERATING_TEMPERATURE,FORE=>2,AFT=>4,EXP=>0);
PUT("    deg kelvin");

SUMMER_SOLSTICE_OPERATING_TEMPERATURE:=((EFFECTIVE_SOLAR_CELL_ABSORBTANCE
   *FRONT_ARRAY_AREA
   *SOLAR_INTENSITY_SUMMER_SOLSTICE
   *COS(SOLAR_ASPECT_COEFFICIENT_SOLSTICE*PI/180.0)) -- 0.0 degrees = 1.0

   /((CELL_EMITTANCE_FRONT*FRONT_ARRAY_AREA+CELL_EMITTANCE_BACK
   *BACK_ARRAY_AREA)*STEFAN_BOLTZMANN))**0.25;

NEW_LINE(2);
PUT("Summer Solstice Operating Temperature is ");
SET_COL(55);
PUT(SUMMER_SOLSTICE_OPERATING_TEMPERATURE,FORE=>2,AFT=>4,EXP=>0);
PUT("    deg kelvin");

VERNAL_EQUINOX_OPERATING_TEMPERATURE:=((EFFECTIVE_SOLAR_CELL_ABSORBTANCE
   *FRONT_ARRAY_AREA
   *SOLAR_INTENSITY_VERNAL_EQUINOX
   *COS(SOLAR_ASPECT_COEFFICIENT_EQUINOX*PI/180.0)) -- 0.0 degrees = 1.0

   /((CELL_EMITTANCE_FRONT*FRONT_ARRAY_AREA+CELL_EMITTANCE_BACK
   *BACK_ARRAY_AREA)*STEFAN_BOLTZMANN))**0.25;

NEW_LINE(2);
PUT("Vernal Equinox Operating Temperature is ");
SET_COL(55);
PUT(VERNAL_EQUINOX_OPERATING_TEMPERATURE,FORE=>2,AFT=>4,EXP=>0);
```

```
  PUT("   deg kelvin");


AUTUMNAL_EQUINOX_OPERATING_TEMPERATURE:=((EFFECTIVE_SOLAR_CELL_ABSORBTANC
E
    *FRONT_ARRAY_AREA
    *SOLAR_INTENSITY_AUTUMNAL_EQUINOX
    *COS(SOLAR_ASPECT_COEFFICIENT_EQUINOX*PI/180.0)) -- 0.0 degrees = 1.0

    /((CELL_EMITTANCE_FRONT*FRONT_ARRAY_AREA+CELL_EMITTANCE_BACK
    *BACK_ARRAY_AREA)*STEFAN_BOLTZMANN))**0.25;

  NEW_LINE(2);
  PUT("Autumnal Equinox Operating Temperature is ");
  SET_COL(55);
  PUT(AUTUMNAL_EQUINOX_OPERATING_TEMPERATURE,FORE=>2,AFT=>4,EXP=>0);
  PUT("   deg kelvin");
  NEW_LINE(4);



else
  -- AREA FOR A SPIN STABILIZED SPACECRAFT
  NEW_LINE(2);
  PUT("Please enter the FRONT solar cell array area in meters squared ");
  NEW_LINE(4);
  GET_DATA(FRONT_ARRAY_AREA);
  NEW_LINE(3);
  PUT("Front Array Area is ");
  set_col(55);
  PUT(FRONT_ARRAY_AREA,FORE=>3,AFT=>3,EXP=>0);
  PUT("  m^2");

  VIDEO.CLEAR_SCREEN;


  WINTER_SOLSTICE_OPERATING_TEMPERATURE:=((EFFECTIVE_SOLAR_CELL_ABSORBTANCE
    *SOLAR_INTENSITY_WINTER_SOLSTICE
    *COS(SOLAR_ASPECT_COEFFICIENT_SOLSTICE*PI/180.0))
    /(CELL_EMITTANCE_FRONT*PI*STEFAN_BOLTZMANN))**0.25;

  NEW_LINE(2);
  PUT("Winter Solstice Operating Temperature is ");
  SET_COL(55);
  PUT(WINTER_SOLSTICE_OPERATING_TEMPERATURE,FORE=>2,AFT=>4,EXP=>0);
  PUT("   deg kelvin");

  SUMMER_SOLSTICE_OPERATING_TEMPERATURE:=((EFFECTIVE_SOLAR_CELL_ABSORPTANCE
    *SOLAR_INTENSITY_SUMMER_SOLSTICE
    *COS(SOLAR_ASPECT_COEFFICIENT_SOLSTICE*PI/180.0)) -- 0.0 degrees = 1.0
    /(CELL_EMITTANCE_FRONT*PI*STEFAN_BOLTZMANN))**0.25;
  NEW_LINE(2);
  PUT("Summer Solstice Operating Temperature is ");
  SET_COL(55);
  PUT(SUMMER_SOLSTICE_OPERATING_TEMPERATURE,FORE=>2,AFT=>4,EXP=>0);
  PUT("    deg kelvin");

  VERNAL_EQUINOX_OPERATING_TEMPERATURE:=((EFFECTIVE_SOLAR_CELL_ABSORBTANCE
    *SOLAR_INTENSITY_VERNAL_EQUINOX
    *COS(SOLAR_ASPECT_COEFFICIENT_EQUINOX*PI/180.0)) -- 0.0 degrees = 1.0
    /(CELL_EMITTANCE_FRONT*PI*STEFAN_BOLTZMANN))**0.25;
  NEW_LINE(2);
  PUT("Vernal Equinox Operating Temperature is ");
  SET_COL(55);
  PUT(VERNAL_EQUINOX_OPERATING_TEMPERATURE,FORE=>2,AFT=>4,EXP=>0);
  PUT("    deg kelvin");


AUTUMNAL_EQUINOX_OPERATING_TEMPERATURE:=((EFFECTIVE_SOLAR_CELL_ABSORBTANC
E
    *SOLAR_INTENSITY_AUTUMNAL_EQUINOX
    *COS(SOLAR_ASPECT_COEFFICIENT_EQUINOX*PI/180.0)) -- 0.0 degrees = 1.0
    /(CELL_EMITTANCE_FRONT*PI*STEFAN_BOLTZMANN))**0.25;
  NEW_LINE(2);
  PUT("Autumnal Equinox Operating Temperature is ");
  SET_COL(55);
  PUT(AUTUMNAL_EQUINOX_OPERATING_TEMPERATURE,FORE=>2,AFT=>4,EXP=>0);
  PUT("    deg kelvin");
  NEW_LINE(3);


end if;
  NEW_LINE(OUTATC,2);
  PUT(OUTATC,"Front Array Area is ");
  set_col(OUTATC,55);
  PUT(OUTATC,FRONT_ARRAY_AREA,FORE=>4,AFT=>2,EXP=>0);
  PUT(OUTATC,"  m^2");

  NEW_LINE(OUTATC,2);
  PUT(OUTATC,"Back Array Area is ");
  set_col(OUTATC,55);
  PUT(OUTATC,BACK_ARRAY_AREA,FORE=>4,AFT=>2,EXP=>0);
  PUT(OUTATC,"  m^2");

  NEW_LINE(OUTATC,2);
  PUT(OUTATC,"Winter Solstice Operating Temperature is ");
  SET_COL(OUTATC,55);
  PUT(OUTATC,WINTER_SOLSTICE_OPERATING_TEMPERATURE,FORE=>4,AFT=>2,EXP=>0);
  PUT(OUTATC,"    deg kelvin");

  NEW_LINE(OUTATC,2);
  PUT(OUTATC,"Summer Solstice Operating Temperature is ");
  SET_COL(OUTATC,55);
```

```
PUT(OUTATC,SUMMER_SOLSTICE_OPERATING_TEMPERATURE,FORE=>4,AFT=>2,EXP=>0);
  PUT(OUTATC,"   deg kelvin");


  NEW_LINE(OUTATC,2);
  PUT(OUTATC,"Vernal Equinox Operating Temperature is ");
  SET_COL(OUTATC,55);
  PUT(OUTATC,VERNAL_EQUINOX_OPERATING_TEMPERATURE,FORE=>4,AFT=>2,EXP=>0);
  PUT(OUTATC,"   deg kelvin");

  NEW_LINE(OUTATC,2);
  PUT(OUTATC,"Autumnal Equinox Operating Temperature is ");
  SET_COL(OUTATC,55);

PUT(OUTATC,AUTUMNAL_EQUINOX_OPERATING_TEMPERATURE,FORE=>4,AFT=>2,EXP=>0)
;
  PUT(OUTATC,"   deg kelvin");

end SOLAR_ARRAY_TEMPERATURE;

begin

  CREATE(OUTATC,NAME=>"ARRAYTC.DAT");

  PRINT_HEADER;

  DUAL_SPIN                  (DRUM_SPINNER);

  SOLAR_ARRAY_TEMPERATURE      (DRUM_SPINNER);

  CLOSE(OUTATC);

   STOP;
   NEW_LINE(2);
   PUT_LINE("DATA FOR THIS DESIGN RUN IS LOCATED IN THE FOLLOWING FILE");
   NEW_LINE(2);
   PUT_LINE("                    ARRAYTC.DAT ");
   NEW_LINE(2);
   PUT_LINE("TO KEEP DATA FROM BEING ERASED ON NEXT RUN");
   PUT_LINE("USE DOS COMMAND REN (RENAME) ");
   NEW_LINE(2);
   PUT_LINE("EXAMPLE - REN ARRAYTC.DAT   ARRAYTC.INI");
   PUT_LINE("The .INI could be your initials");

end ARRAY_THERMAL_CONTROL;
```

E. UTILITY SUBPROGRAMS

```
-- Title         : GET DATA
-- Author        : David Lashbrook
-- Date          : 15 February 1992
-- Revised       . 30 March 1992
-- Compiler      : OPENADA EXT
-- Description   : Package gets data for floats, integers, characters

package GETDATA is
  procedure GET_DATA(X : out FLOAT);
  procedure GET_INTEGER(I : out INTEGER);
  procedure STOP;
  procedure GET_CHARACTER(CHAR : out CHARACTER);
end GETDATA;
-----------------------------------------------------------------------------------------
-- Author        : David Lashbrook
-- Date          : 15 February 1992
-- Revised       : 30 March 1992
- Compiler       : OPENADA EXT
-- Description   : Package Body gets data for floats, integers, characters

with TEXT_IO, MATH_LIB, VIDEO;
use  TEXT_IO ;

package body GETDATA is

  package FLOAT_INOUT is new FLOAT_IO(FLOAT);
  use     FLOAT_INOUT;
  package INTEGER_INOUT is new INTEGER_IO(INTEGER);
  use     INTEGER INOUT;

  X                    : FLOAT ;

  CHAR                   : CHARACTER ;

  I                    : INTEGER ;




procedure GET_DATA(X : out FLOAT) is
  begin
    loop
      begin
          SET_COL(10);
          PUT_LINE("Enter the value as a real number with a decimal point");
          SET_COL(15);
          PUT_LINE("(Depress CTRL^C to exit the program.)");
          SET_COL(10);
          GET(X);
          SKIP_LINE;
          exit;
       exception
          when DATA_ERROR =>
            SKIP_LINE;
            NEW_LINE;
            SET_COL(10);
            PUT_LINE("Error.. You must enter the value as a real");
            SET_COL(10);
            PUT_LINE("number with a decimal point.  ie 123.4");
            SET_COL(10);
            PUT_LINE("Try again.");
            NEW_LINE;
         end;
   end loop;
end GET_DATA;

-- Reads an integer input from the keyboard

 procedure GET_INTEGER(I : out INTEGER) is
 begin
   loop
          begin
            NEW_LINE(1);
            SET_COL(10);
            PUT_LINE("Enter the value as an integer");
            PUT_LINE("(Depress CTRL^C to exit the program.)");
            SET_COL(10);
            GET(I);
            SKIP_LINE(1);
            exit;
          exception
            when DATA_ERROR =>
              SKIP_LINE;
              NEW_LINE;
              SET_COL(10);
              PUT_LINE("Error.. You must enter the value as a INTEGER");
              SET_COL(10);
              PUT_LINE(" NO!  decimal point.  ie 123 ");
              SET_COL(10);
              PUT_LINE(" Please try again.");
              NEW_LINE;
          end;
   end loop;
 end GET_INTEGER;

procedure STOP is

N      : INTEGER ;
begin
```

```
  SET_COL(10);
  PUT("TO CONTINUE ENTER ANY INTEGER");
  GET_INTEGER(N);
  VIDEO.CLEAR_SCREEN;
end STOP;


procedure GET_CHARACTER(CHAR : out CHARACTER) is
 begin
  loop
        begin
         SET_COL(10);
         PUT_LINE("Enter  'Y'  for YES or  ");
         NEW_LINE(1);
         SET_COL(10);
         PUT_LINE("       'N'  for NO");
         SET_COL(15);
         PUT_LINE("(Depress CTRL^C to exit the program.)");
         SET_COL(10);
         GET(CHAR);
         SKIP_LINE;
         exit;
        exception
         when DATA_ERROR =>
         SKIP_LINE;
         NEW_LINE;
         SET_COL(10);
         PUT_LINE("Error.. You must enter character");
         SET_COL(10);
         PUT_LINE("Try again.");
         NEW_LINE;
        end;
  end loop;
 end GET_CHARACTER;


end GETDATA;
```

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	(2)
2.	Library, code 0142 Naval Postgraduate School Monterey, California 93943-5002	(2)
3.	Professor Brij N. Agrawal Halligan Hall, Code AA/Ag Naval Postgraduate School Monterey, California 93943	
4.	CDR. Randy Wight Bullard Hall, Code SP/Wt Naval Postgraduate School Monterey, California 93943	
5.	LCDR David Lashbrook 2801 Ringgold Court Woodbridge, Virginia 22192	
6.	Aeronautics and Astronautics Department Halligan Hall, Code AA Naval Postgraduate School Monterey, California 93943	